3

Theory of Constraints

Theory of Constraints (TOC) is one of the best-known management approaches you may never have used, even if you're a professional manager or executive. The most widely acclaimed book about TOC has sold millions of copies, and its lessons have been widely influential in industry.[1] Several dozen other books and hundreds of articles about TOC have since been published. Nevertheless, far more TOC references apply to industry than services. So if you're in a services sector, TOC may not be on anybody's agenda yet. That's understandable, because TOC was invented to address chronic problems in industry. From its origin in operations, however, TOC has been extended into other business functions in industry, including distribution, engineering, finance, marketing, sales, strategy, and change management. Moreover, it has been adopted occasionally in various services enterprises. This chapter briefly surveys TOC applications for industry because that foundation will be helpful in understanding how and why the TOC applications for services in this book are different.

There's also a branch of TOC known as the *Thinking Process*, which is applicable in any problem-solving situation. Though this book is a result of that Thinking Process, there's nothing about the Thinking Process itself that requires special adaptation for services. Thus, no chapters in this book are devoted to it. If you're interested, however, it's covered in most other TOC books, including some that are dedicated to the Thinking Process.[2,3]

In published cases of TOC usage in services, a common approach is to start with the Thinking Process and then use it to figure out which TOC applications from industry might apply. Here are some of the best-documented cases of TOC in services:

- Performing agricultural services[4]
- Changing the hiring process in a police department[5]
- Scheduling patient services in a hospital[6]
- Managing software engineering[7]

In each case, the authors were able to apply TOC because they were dealing with services delivered via relatively repeatable processes.

Software engineering is arguably the least repeatable. Yet unlike pure services, software inventories must be managed, so the software business is more like manufacturing in that way.

The rest of this chapter covers standard TOC applications. The terminology and examples come from industry, but each application has also been used in services to varying degrees. If you're already familiar with TOC, you may want to skip this chapter or skim selected sections because the coverage is more broad than deep. Conversely, if TOC is new to you, you may also want to explore the endnotes for more information. In either case, be aware that this chapter lays the foundation for the adaptation of TOC for Services in subsequent chapters.

All TOC applications spring from a common premise: If an enterprise is viewed as a chain, the enterprise as a whole can produce only as much as its weakest link will allow. That weakest link is, of course, the constraint. But the chain analogy has another implication: Pulling a chain is a lot more effective than pushing it. So switching enterprises from push to pull is a key ingredient in every TOC application.

Drum-Buffer-Rope

Drum-Buffer-Rope (DBR) is the TOC application for operations.[8] It's often used to plan and manage discrete manufacturing, but DBR has also been used by service providers as diverse as landscapers and hospitals, even though their services are perishable.

DBR gets its name from the roles that specific elements play during scheduling and management of production. To appreciate those roles, it helps to know the problems DBR was originally intended to solve. For that we need a quick review of the state of the art in manufacturing when DBR was invented.

A simplified manufacturing process composed of just five steps is illustrated in Figure 3-1. Of course, actual manufacturing processes are often composed of many steps in complex configurations shaped like a V, A, T, I, or some combination thereof. But a simple sequence of steps is sufficient to illustrate the essential elements of DBR.

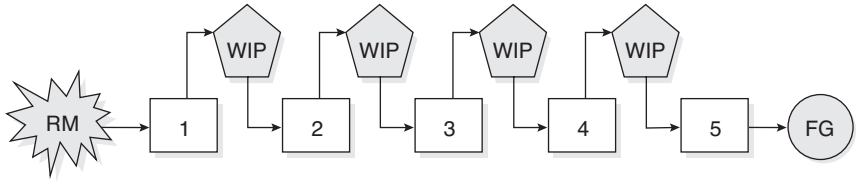

Figure 3-1 Traditional manufacturing

The steps are numbered 1 through 5. Assume that each step in this illustration is performed by a different machine type. RM stands for raw materials, WIP stands for work in process, and FG stands for finished goods, which are all forms of inventory.

One problem with traditional manufacturing is immediately apparent in the figure: There's a lot of inventory before, during, and after production. That's a problem because inventory is a significant investment, and it doesn't generate revenue until it's sold.

Another problem is that excess inventory impedes the production process. That is, as the shop floor becomes crowded with WIP, it gets harder to monitor due dates and ensure that the most urgent jobs are done first. The busier the shop gets, the less effective expediting becomes.

Thus, a third problem is that it's hard to predict when each job will be completed. Once jobs are released into the shop, they are hard to control. Some jobs may finish early, but too many finish late, which leads to customer dissatisfaction and missed sales. So as production slows, jobs may be started earlier, thereby further increasing WIP, slowing production, and perpetuating the push cycle.

Work also gets pushed into and through the factory by the desire for high utilization, a measure of how long each machine and each worker are actually performing tasks in the production process. The underlying assumption is that anything less than high utilization on every machine and every worker represents a lost opportunity for production.

Though appealing, that assumption is flawed. For one thing, producing goods that customers won't buy is wasteful, no matter how high it drives utilization. Yet even when customers will gladly buy what's produced, the push for universally high utilization overwhelms the constraint.

Somewhere in that production process is a step that cannot produce as many units per time period as the rest of the steps. That's the constraint. You can't see it in the figure, and neither can most managers in an actual manufacturing plant managed the traditional way. Fortunately, the constraint is hiding in plain sight, and with a little detective work it can be found.

What's far harder to do is change the perception that high utilization everywhere is a good thing. The belief that local optimizations somehow add up to global optimization is strongly held. Until this policy constraint is broken, however, the physical constraint cannot be managed.

The same five-step manufacturing process as before is illustrated in Figure 3-2, but it also includes the elements needed for DBR. Solid lines represent product flow. Dashed lines represent information flow. In an actual factory with many products and a variety of routings, there can be more than one constraint. To illustrate DBR, however, one will do.

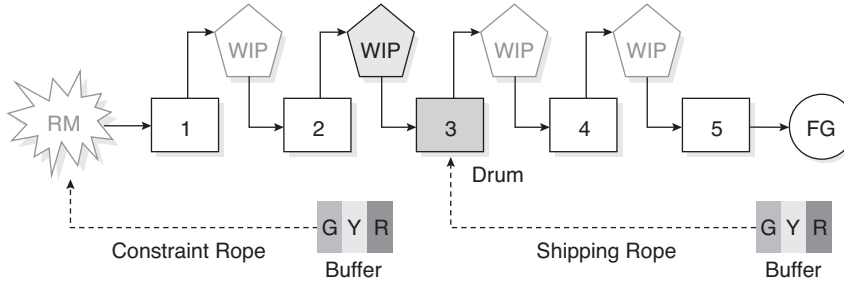

Figure 3-2 Drum-Buffer-Rope (DBR)

Step 3 is the constraint. It's also called the drum because it sets the pace for the rest of the steps. That is, upstream steps will occasionally be idle—have less than full utilization—so they won't overwhelm the constraint with work. And downstream steps will likewise occasionally be idle because they're waiting for the constraint to complete its step. But if all goes well, the constraint itself will have consistently high utilization, excess WIP will disappear, and more orders will ship on time.

When this happens, the factory is producing as much as it can, subject to the current constraint. This level of production is typically much more than it ever could produce under traditional manufacturing when

the constraint was invisible. So managers in a factory adopting DBR may go from wrestling with insufficient capacity to having ample capacity.

A likely place to see WIP is ahead of the constraint, because it can produce less than any other step, by definition. Therefore, that WIP is sometimes mistaken for the buffer, but the drum buffer is actually all work scheduled on the constraint, even if it's currently at an earlier step. That is, the buffer is measured in time, not physical WIP units. So a view of the true buffer is typically contained in an information system nowadays.

If all jobs ahead of the constraint are early or on schedule, the amount of work needed to keep the constraint busy is adequate, and the buffer is said to be in the green zone. However, when some jobs are behind schedule and the possibility that the constraint could run out of work becomes significant, the buffer is in the yellow zone.

Because normal variation causes some jobs to run early at the same time that others run late, it's possible that the constraint won't actually run out of work. Hence, a yellow buffer does not automatically trigger action. However, when many jobs are behind schedule and it becomes clear that the constraint will indeed run out of work without action, the buffer is in the red zone. Upstream steps then have to sprint to refill the buffer, and thereby keep the constraint busy, while downstream steps may have to sprint to finish late jobs on time.

In addition to the drum buffer, which contains WIP, the shipping buffer contains FG. Because the market is the ultimate pacesetter, the shipping buffer protects customers from late delivery, just as the drum buffer protects the constraint from overloading.

The third and final element of DBR is the ropes, which govern when gating events occur. The shipping rope governs work on the constraint needed to meet market demand and keep the shipping buffer green. The constraint rope governs the release of raw materials to start new jobs that should keep the drum buffer green.

Under DBR, jobs are released much closer to their due date than in traditional manufacturing because they will spend less time waiting between steps. Like the buffer, the length of ropes is measured in time, and the ropes are actually contained in an information system.

An information system that supports DBR leads to global optimization by optimizing the constraint rather than every step in the production process.[9] Buffer management thus keeps the process in control without requiring constant attention and fine-tuning. If a factory has more than enough capacity to meet market demand, the constraint is said to be *external* or *in the market*. In this case, the shipping buffer, not the drum buffer, then regulates when jobs are released into the shop because the internal constraint no longer limits production. If a factory has enough capacity to meet market demand during normal and slack periods but not during peak periods, its dominant constraint is in the market even though it occasionally has an internal constraint.[10] In this case, a simplified form of DBR can be implemented that makes sensible trade-offs between keeping the constraint busy and satisfying customer demand in order to protect future sales. In both cases, when the constraint is external, no step has full utilization, including the internal constraint, but this is what keeps the factory from producing excess inventory that cannot be sold. The next section covers another TOC application that specifically addresses a market constraint.

Whenever the constraint shifts (due to changes in machines, people, process, products, or demand), DBR has to be reconfigured accordingly. This is a nontrivial effort, and it's why capacity in a DBR shop is deliberately unbalanced to prevent floating constraints.

Placement of the drum, however, should be strategic, not accidental. That is, when DBR is first implemented, the constraint location may not be correctly aligned with respect to profitable market opportunities. If so, rather than implement DBR around this previously unseen constraint, it generally makes more sense to adjust capacity so that the control point represented by the drum is relocated to a position where the factory will be better able to produce goods that meet market demand profitably.

DBR is also known as *Synchronous Manufacturing*. In a nutshell, here's how it compares to two other widely used production management approaches:

- In their pure forms, Enterprise Resource Planning (ERP) assumes infinite capacity and schedules all steps, while DBR assumes finite capacity and schedules just the constraint. Some ERP software can

schedule to finite capacity, but it does not have other essential capabilities of DBR software. For instance, ERP prohibits late release of materials, while DBR prohibits early release because it increases WIP. Moreover, ERP drives material requirements all the way through the bill of materials (BOM), regardless of stock on hand, while DBR takes existing stock and buffers into consideration. Thus, ERP and DBR are fundamentally different solutions.

- Lean/Just-in-Time (JIT) seeks to optimize individual steps, while DBR optimizes the entire process around the constraint. They are fundamentally similar, but Lean/JIT doesn't work as well in job shops as flow shops because job shops have more diverse and changeable routings.

The benefits of DBR are substantial. One literature review found the following average improvements across 82 companies:[11]

- 70 percent reduction in lead time
- 65 percent decrease in cycle time
- 44 percent improvement in due-date performance
- 49 percent reduction in inventory
- 63 percent increase in revenue

A central benefit of DBR is to change the production process from push to pull: Nothing gets produced unless there's a market for it. Market pull through the internal constraint then optimizes production while minimizing inventory.

Because the market is the key driver of DBR, how demand ripples back through the distribution chain from customers to factory affects DBR. This connection leads to the next TOC application.

Replenishment

Replenishment (R) is the TOC application for distribution.[12] It was originally invented to manage distribution of goods, but it can be used by service providers who deliver goods along with their services, such as those in the Accommodations and Food Services sectors.

Replenishment is also called the TOC supply chain solution because one enterprise's distribution chain is often another's supply chain. Of

course, calling them "chains" is not entirely accurate because they are increasingly complex networks. Nevertheless, where one enterprise's distribution network and interlocks with another enterprise's supply network, they are essentially the same elements viewed from different perspectives.

Replenishment gets its name from the specific manner in which goods are distributed or supplied. As with the previous TOC application, it is easier to appreciate Replenishment in the context of the problems it was originally intended to solve. Hence, we'll do another quick review of the state of the art, only this time it'll be in distribution, as of the invention of Replenishment.

A simplified distribution chain is illustrated in Figure 3-3. A real enterprise might have far more factories, warehouses, and retail outlets in its distribution chain, of course. And a real supply chain can look like a mirror image of this figure, with many sources funneling into a central location. Furthermore, the individual elements might be owned by a single enterprise or many different enterprises. Although it can be harder to implement a common solution across enterprises, Replenishment itself doesn't depend on who owns each element. Thus, this simple scenario is sufficient to illustrate Replenishment.

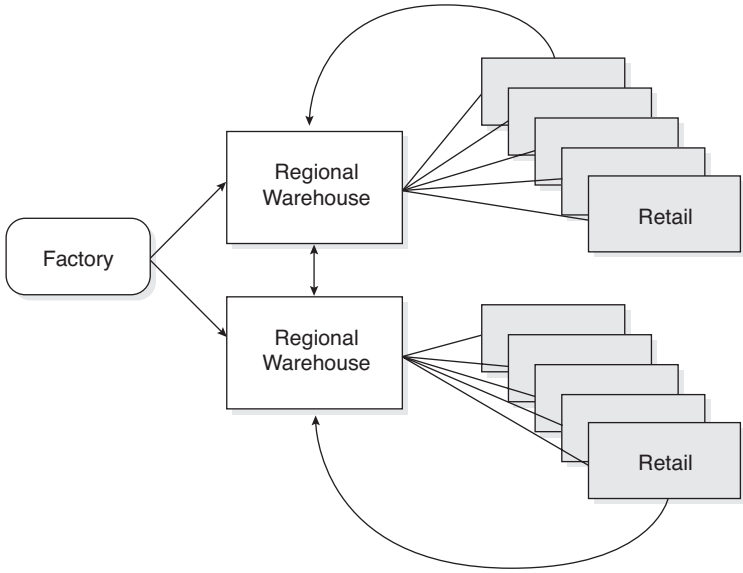

Figure 3-3 Traditional distribution

With traditional distribution, goods produced by the factory are immediately shipped in large batches to regional warehouses. Each regional warehouse in turn periodically ships smaller but still sizable batches to retail locations. Thus, most inventory is pushed through the chain to retail locations on the assumption that it will eventually be sold. This method is intuitively appealing, because only retail outlets make sales to consumers.

Unfortunately, variability in sales is highest at retail locations. This has several undesirable effects. First, some retailers may have an abundance of some products at the same time that others have none. Second, there's seldom an easy way to move inventory between retail locations to reduce overstocks and cover stockouts. Finally, when stockouts occur, if the time required to restock a retailer from the warehouse is longer than customers will wait, stockouts turn into lost sales, not backorders.

Thus, retailers with excess inventory return it to the warehouse, while those with no inventory lose sales while waiting for shipments. And when cross-shipments between warehouses are needed to cover shortages, those shipments may be delayed by the desire to ship large batches in order to save shipping costs.

Furthermore, whenever new products are introduced, retail locations—if not the entire distribution chain—tend to be filled with old products. So as new products are pushed through the chain, the old products must be discounted to clear them from retailers' inventory. And those discounts cut into sales of the new products, too.

The size and timing of batches pushed through the chain depend mainly on whether the factory follows traditional manufacturing or DBR/Lean/JIT. Traditional manufacturing is driven by sales forecasts. Yet when products are pushed through a traditional distribution chain, sales forecasts are notoriously inaccurate. So a vicious cycle is at work: The bigger the batches and the less frequently they're distributed, the longer the horizon on sales forecasts, which in turn makes forecasts even less accurate, which calls for bigger batches, and so forth.

Therefore, the net result of inaccurate forecasting and pushing large batches through a distribution chain is low reliability even when the chain is filled with excess inventory. This is another example of local optimization not leading to global optimization.

As in the previous section, the constraint may not be immediately obvious, but it's hiding in plain sight. It may be tempting to conclude that transportation or warehouse capacity is the constraint, but because inventory tends to pile up ahead of the true constraint, that's the clue we

need. The amount of product that should be distributed is ultimately limited by sales to customers, so the constraint in a distribution chain is usually external or in the market.

The best way to break a sales constraint is simply to sell customers what they want, when and where they want it, at a price that corresponds with perceived value. Hence, to get the right products in the right amounts to the right locations at the right time, the solution has to be an alternative to sales forecasts, big batches, and infrequent shipments. Indeed, Replenishment turns traditional distribution completely on its head by eliminating sales forecasts and shipping small orders quite frequently.

The same distribution chain as before is illustrated in Figure 3-4, but it also includes the elements needed for Replenishment. Solid lines represent product flow. Dashed lines represent information flow.

Rather than being located only adjacent to the constraint, however, the biggest Replenishment buffer is located at a better leverage point. That is, most inventory is held in the factory warehouse buffer for reasons discussed next. Only a single factory warehouse buffer is shown in the figure for illustration, but each product actually has its own buffer. Likewise, each regional warehouse and retail location have buffers for each product.

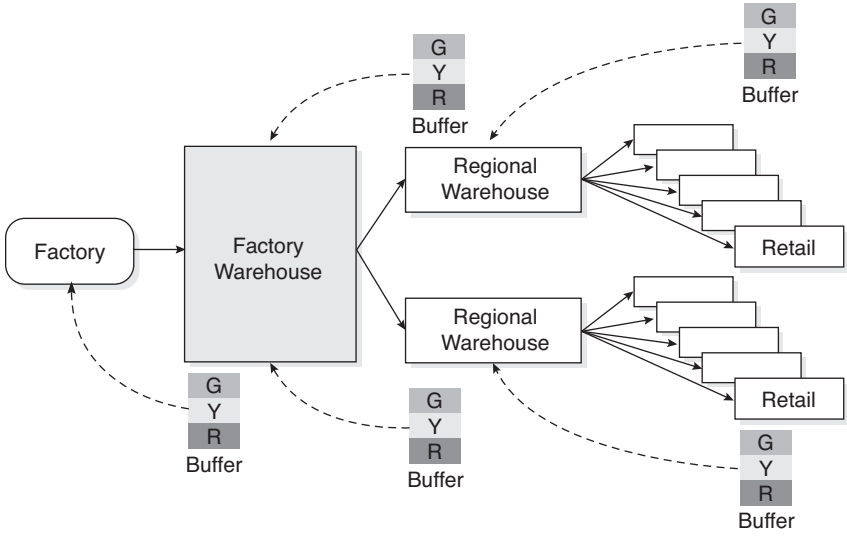

Figure 3-4 Replenishment

Buffers are divided into green, yellow, and red zones that correspond to no action, replenish, and expedite. But rather than being measured in time periods as in DBR, Replenishment buffers are measured in physical units of inventory.

Replenishment relies on *aggregation* to smooth demand. That is, demand at regional warehouses is smoother than demand at retail locations because higher-than-normal demand at some retail locations is offset by lower demand at other ones. Likewise, demand at the factory is even smoother than demand at the regional warehouses. Therefore, goods produced by the factory are stored in a nearby warehouse and are not shipped elsewhere until they are needed to replenish goods actually consumed by sales. The factory warehouse may be owned by a distributor rather than the manufacturer, but the effect is the same.

Because sales occur daily, shipments occur daily, too. And the quantities shipped are just sufficient to replace goods sold. At first glance this might seem to increase shipping costs over what could be achieved by shipping large batches less frequently, but the net effect on total shipping costs is that they usually go down, not up. Stopping the shipment of obsolete goods and reshipment of misallocated goods more than compensates for increased cost created by smaller shipments of saleable goods. Furthermore, the ability to capture sales that would otherwise be lost due to insufficient inventory makes Replenishment a much better alternative.

Thus, Replenishment is driven by actual consumption, not sales forecasts. As sales are made, the buffer levels at retail locations drop, eventually triggering Replenishment from a regional warehouse. Similarly, as buffer levels at regional warehouses drop, this eventually triggers Replenishment from the factory warehouse. The buffer zones at the factory warehouse are set, however, so that they trigger a manufacturing order that should resupply this buffer before it runs out.

Buffer sizing is based on both variability and time to resupply. That is, the more variable consumption is, the bigger the buffer must be to cover that variability. Likewise, the longer it takes to resupply, the bigger the buffer must be to cover demand during the wait. Therefore, as aggregation reduces variability and DBR reduces resupply time, the required buffer size decreases accordingly.

In addition to being used subsequent to DBR, Replenishment can be combined with DBR. That is, the Replenishment techniques that work in an external distribution chain can also be used internally within a

manufacturing plant. This is appropriate when the same materials, parts, or subassemblies are used in multiple products. Rather than keeping separate buffers of the common items for each product being manufactured, keeping a central buffer and replenishing common items as they are consumed leads to higher reliability with less total inventory.

Furthermore, strategically placed replenishment buffers enable a manufacturer to reduce end-to-end production schedules. DBR software with replenishment capabilities is required, however, because ERP systems drive requirements all the way through the BOM, regardless of stock on hand or replenishment buffers.

If a manufacturer's business is predominantly design-to-order or make-to-order, using Replenishment to distribute its primary products is not as appropriate. However, even in those types of manufacturing, spare parts are usually make-to-stock, and Replenishment is quite applicable to them. Because spare-parts inventory for a large enterprise often represents an investment of hundreds of millions of dollars, a significant reduction in that inventory frees capital for reinvestment in other areas.

Several Replenishment outcomes are noteworthy:

- Reliability is significantly increased by replenishing goods based on actual consumption.
- Inventory is substantially decreased by keeping the bulk of it where demand varies least.
- Sales can be significantly increased by reducing delivery time and eliminating stockouts.
- Shipping large batches to save shipping costs is false economy.

The benefits of Replenishment can be striking. A traditional distributor that is 85 percent reliable can increase its reliability to 99 percent while reducing its inventory by two-thirds when adopting Replenishment. Furthermore, the average time to resupply retail locations typically drops from weeks or months to about one day.

A central benefit of Replenishment is thus to change the distribution chain from push to pull: Nothing gets distributed unless there's a market for it. Market pull, the external constraint, then optimizes distribution while minimizing inventory.

As seen in the previous section, when DBR unleashes latent manufacturing capacity, the enterprise's constraint often moves from internal to external. Conversely, when Replenishment unleashes latent distribution capacity, if there is sufficient demand, the constraint can move back from external to internal. The internal location may be engineering of new products rather than manufacturing, however. This connection leads to the next TOC application.

Critical Chain

Critical Chain (CC) is the TOC application for project management.[13] It was originally invented to manage engineering projects in a manufacturing environment, but it has since been used by enterprises in virtually all goods and services sectors.

Critical Chain gets its name from the specific manner in which projects are planned and executed. As with the previous TOC applications, it is easier to appreciate Critical Chain in the context of the problems it was originally intended to solve. Some aspects of Critical Chain have been widely adopted and are no longer distinctive, but most still struggle to gain acceptance. The following review of the state of the art compares Critical Chain to the older and still dominant project management method, *Critical Path*.

A simple project plan based on the Critical Path method is illustrated in Figure 3-5. Each block represents one task. The width of each block against the timeline at the bottom shows its planned duration. Adjacent blocks have a finish-start relationship: The task on the left must be finished before the task on the right can start. Where more than one task must be completed before another can start, this precedence relationship is shown with arrows. Precedence arrows between adjacent blocks are implied rather than explicit.

Each task is numbered, but the numbers are merely identifiers. They do not imply sequence. Subscripts on the task numbers indicate which resource is assigned to each task. Assume in this example that only one resource is assigned to each task.

Red tasks are critical, which means that if any of them are completed late, the entire project will be late unless some other critical tasks are completed early. Yet early task completions rarely happen. Thus, the set of all red tasks is the critical path.

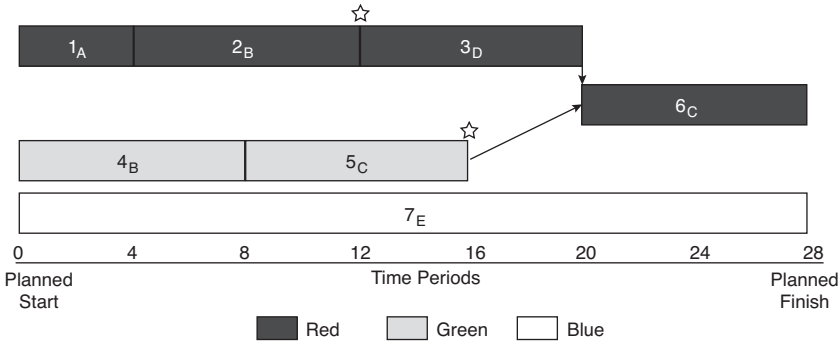

Figure 3-5 Critical Path

Green tasks are noncritical, which means that they have slack time. *Slack* is represented by the horizontal component of the arrow to their left or right. If they exceed this slack time, they too will make the project late. Hence, they are pushed leftward to indicate they will start as early as possible in order to preserve the slack. Task 5 could finish as early as Period 16 or as late as Period 20, and the project overall could still finish on time.

Blue tasks are *elastic*, which means they are based on the duration of nonblue tasks, so they expand and contract with the rest of the project. Thus, even if they have no slack, they cannot make the project exceed its planned duration. They can, however, make the project start late if an appropriate resource is not available. Project management itself is an example of an elastic task. Such tasks are usually omitted from plans for internal projects because they don't affect the critical path, but they are often included in plans for services projects to make project managers' and partners' tasks visible to the client.

Stars represent *milestones*, which are significant events or deliverables. Milestones are often used to mark the end of project phases, and thus serve as interim progress indicators. Some methodologies require at least one milestone per month.

Real projects generally have many more tasks and a web of precedence relations, of course. Nonetheless, even this simple project is sufficient to compare Critical Chain to Critical Path.

In contrast to previous TOC applications where the constraint was hidden, the constraint in this project plan may seem obvious. The critical path appears to be the constraint because it determines the shortest

time in which the project can be completed if all goes according to plan. However, resources are a far less obvious constraint hiding in plain sight on many project plans. For example, unless people examine a separate resource view of the plan, many don't notice that two tasks in this project must be performed simultaneously by Resource B.

Frequent shifting between multiple tasks in order to create the appearance of simultaneous execution is called *multitasking*. A widespread assumption is that it increases productivity, but this assumption is faulty for a couple of reasons. First, it takes time to switch between tasks. More significantly, however, resources are sometimes assigned more simultaneous tasks than they can possibly complete on time. Bad multitasking occurs when the resulting productivity drain or overload are significant enough to cause tasks to fall behind schedule. This can cause the critical path to shift multiple times during a project, so frequent replanning is common.

In this example, Task 4 has some slack, but not enough to eliminate multitasking by rescheduling that task to its latest start time. If another resource with the same skills as Resource B can be assigned—a technique known as *crashing*—neither task will need to be rescheduled. On the other hand, if no other suitable resource is available, one of the tasks must be rescheduled, and the overall project duration will be longer. *Resource leveling*, as this adjustment is known, is now considered a best practice, even in the Critical Path method, but it was not common when Critical Chain was invented.

Another problem exists in the planned duration of individual tasks. Traditional project management includes a margin for *contingency* in every task estimate because individuals giving those estimates are held accountable if their tasks are completed late. On the other hand, they are also held accountable for the accuracy of their estimates, so they have a disincentive to complete tasks early. Hence, if a project due date isn't chosen arbitrarily—independent of scope—it is most likely based on inflated task estimates.

Despite these margins for contingency embedded throughout, projects too often finish late. One reason is that tasks completed early rarely compensate for tasks completed late. That is, late completions tend to be cumulative. Thus, protecting commitments on every task with contingency does not protect the project due date. This is yet another example of how local optimization everywhere does not create global optimization.

Finally, traditional project management often measures project status according to the percent of the project completed. This measure is misleading, however, because most of a typical project's tasks are not critical, which means they have no bearing on whether the project will actually be completed on time. Hence, projects that reach the 90 percent-completed stage early are often still in danger of finishing late. As the due date approaches, it may be necessary to compromise on one or more of the project commitments by reducing scope, increasing budget (by adding resources), or accepting late completion.

Figure 3-6 illustrates a Critical Chain project plan with the same scope, tasks, resources, and deliverables as before, but it is literally not the same project.

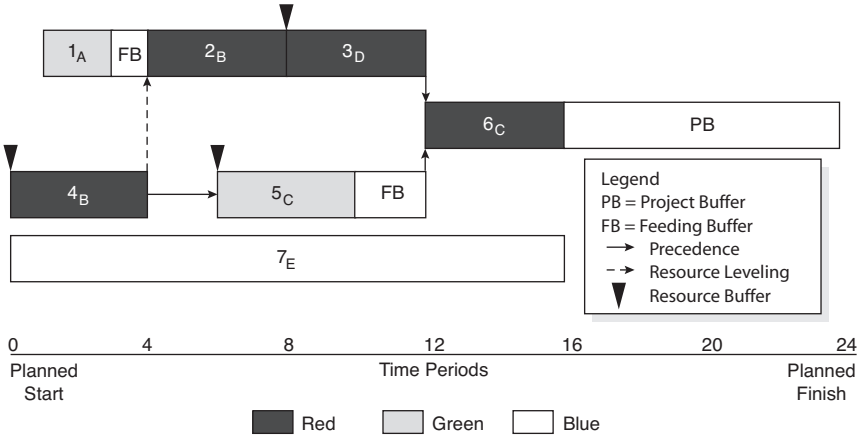

Figure 3-6 Critical Chain

The Critical Chain plan differs from the previous plan in these ways:

- All estimated task durations have been cut in half.
- Resource contention has been resolved via resource leveling.
- Tasks do not start until needed but must be completed as soon as possible once started.
- A project buffer has been added at the end of the project.
- Feeding buffers have been created where noncritical tasks precede tasks on the critical chain.
- Resource buffers have been inserted to remind resources of upcoming tasks.

- The critical chain includes a task not on the critical path, and vice versa.
- Overall project duration is shorter, both with and without the project buffer.

The Critical Chain is the longest set of dependent tasks, taking both precedence and resources into account. If there were no resource contention, the Critical Chain and the Critical Path would include the same tasks. But whenever resource contention exists, as in this example, tasks on the Critical Chain may differ from those on the Critical Path. Furthermore, real projects may have many potential Critical Chains, so computer software is required to find the best resource-leveling alternatives on all but trivial projects.

Even though there is no precedence relation between Tasks 2 and 4 in this example, Task 4 is rescheduled to finish before Task 2 because these tasks are assigned to the same resource. This resource leveling causes Task 4 to become part of the Critical Chain because it now affects overall project duration.

Task durations are cut in half because most individuals give task estimates that they have at least an 80 percent chance of meeting, and a task estimate that's 80 percent reliable is roughly twice as long as one that's 50 percent reliable. In addition to anticipating significant multitasking, an 80 percent estimate anticipates that managers will cut the estimates during planning—and during execution, many tasks will be late even before they start. As you will see, however, these additional concerns are addressed, during both planning and execution of projects.

In the aggregate, task estimates that are 50 percent reliable are sufficient to complete the project on time if there is no bad multitasking, because early task completions can offset late task completions. Furthermore, a portion of the contingency that used to be embedded in the task estimates has been moved into the project buffer, where it now protects the entire project rather than individual tasks.

Feeding buffers protect the Critical Chain. If noncritical tasks are completed late (as they will be about half the time because the plan is based on estimates with 50 percent reliability), the feeding buffers usually prevent accumulated delay from passing into the Critical Chain. Conversely, feeding buffers also allow early completions on the Critical Chain to be immediately followed by early starts on the Critical Chain.

Resource buffers are early reminders to resources with tasks on the Critical Chain that they must start their assigned tasks as soon as previous tasks are complete. That is, an early finish on one task must be followed by an early start on subsequent tasks unless resource contention prevents it.

Note that this project plan has no milestones because they encourage local optimization. Furthermore, because milestones are a commitment to the project sponsor or client, they need their own completion buffers, which could lengthen the project.

Projects managed according to Critical Chain are more likely to finish on time, in part because their status is measured differently. A key measure is the amount of the project buffer used relative to the amount of the Critical Chain completed. For example, if the Critical Chain is 50 percent complete and only 10 percent of the buffer has been used, the project is in excellent shape. But if the Critical Chain is 50 percent complete and 70 percent of the buffer has been used, the project is in danger of consuming the entire buffer and thereby missing its due date.

In general, estimated project duration based on Critical Chain can be up to 25 percent shorter than an equivalent traditional project. However, a traditional project has a high probability of being late, while a Critical Chain project has a high probability of being on time. Therefore, the actual difference in duration can be even larger than this difference in estimates.

Nevertheless, the committed due date in Critical Chain is always the planned finish date at the end of the project buffer, not the finish of the last task on the Critical Chain. This is a crucial distinction because the probability that the last task on the Critical Chain will be completed on time is no better than 50 percent. The probability that the entire project will be completed by the end of the project buffer is better than 90 percent.

A central benefit of Critical Chain is thus to change project management from push to pull: Rather than starting every task as early as possible and pushing every task for on-time completion, Critical Chain starts tasks at just the right time and lets buffer management pull the entire project to on-time completion. Furthermore, better estimating, work rules, and progress measurement optimize due-date performance while minimizing project-wide work effort. In other words, the purpose of Critical Chain is not just to create a better plan, but to change behavior during project execution. Critical Chain projects are thus like a relay

race: Each task on an active chain starts as soon as its predecessors are complete instead of when they were originally planned.

Several Critical Chain outcomes are noteworthy:

- It allows more projects to be completed without increasing staff.
- The Critical Chain is more stable than the Critical Path, so replanning is not as common.
- Buffer penetration provides an unambiguous indicator of whether the project is on schedule.
- Eliminating bad multitasking makes individuals more productive and less prone to burnout.
- Knowing which tasks cause buffer penetration creates a clear priority for project managers.

Project managers must have more than passing familiarity with Critical Chain to use it effectively, but resources working on Critical Chain projects do not need to understand all the details. It is vital, however, that they understand the reasons behind the changes that Critical Chain brings to their jobs. For example, without such understanding, cutting task estimates in half will be perceived as exploitation that creates immense professional risk. On the other hand, understanding that resource leveling reduces overtime, that buffers protect the entire project, and that measures establish clear priorities is a route toward buy-in.

When an enterprise conducts multiple projects with shared resources, resource leveling should extend across projects. Otherwise, cross-project multitasking can increase markedly, thereby endangering timely completion of more than one project. Enterprises facing this challenge are likely to have both resource managers and project managers interacting via matrix management, which is considerably more complicated than a single-project environment and rife with opportunities for conflict.

Critical Chain has been extended to handle multiple projects with shared resources, but the prevailing method does not adequately address constraints in all services enterprises. Therefore, discussion of the multi-project extension will be deferred to a later chapter, where an alternative method for services is compared to the prevailing method that arose in industry.

As explained in Chapter 2, "Services On Demand," as TOC applications are first adopted by an enterprise, constraints can often be identified with a bit of informal detective work. As TOC applications are deployed, however, effective management of constraints requires different kinds of information than the enterprise has ever used before. That is the purpose of the next TOC application.

Throughput Accounting

Throughput Accounting (TA) is the TOC application for finance and accounting.[14] It was originally invented for use in a manufacturing environment, but has since been adapted for use in the computer software industry, as explained in Appendix C, "Throughput Accounting for Software." TA has been the subject of many books and articles, so the collective body of knowledge runs thousands of pages. Only portions relevant to this discussion are summarized in this section, however.

TA gets its name from the specific way financial measures are calculated and used for decision making. As with the previous TOC applications, it is easier to appreciate TA in the context of the problems it was originally intended to solve. The following review of the state of the art compares TA to the older and still dominant management accounting method, *Cost Accounting* (CA).

Figure 3-7 illustrates selected elements of CA. When CA arose in the early 1900s, labor costs dominated manufacturing, and workers were paid by the piece. Hence, it was reasonable at that time to allocate overhead expenses to products on the basis of direct labor costs for purposes of preparing financial statements.

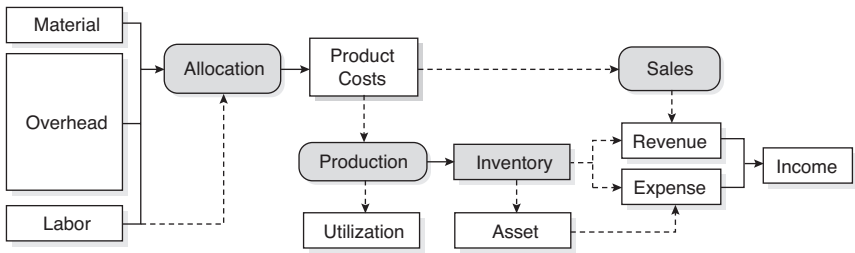

Figure 3-7 Cost Accounting

Ever since automation came to dominate manufacturing and workers came to be paid by the hour, however, allocation of large overhead expenses on the basis of small labor costs has created distortions. When re-aggregated at the enterprise level, product cost distortions do not affect financial statements. Yet if prices are computed as product cost plus standard gross margin, the prevailing method, product cost distortions carry into product pricing. Hence, some products may appear profitable when they are in fact unprofitable—and vice versa.[15]

Activity-Based Costing (ABC) is a relatively recent variant of CA, designed to be a better way to allocate costs. For example, a product manufactured in large batches requires fewer setups and inspections, so it gets allocated less overhead. Unfortunately, ABC does not consider that customers may not want to buy such products.

> ***"Published studies demonstrate an overwhelming tendency for companies to use cost-driver information to do efficiently what they should not be doing in the first place. To achieve competitive and profitable operations in a customer-driver global economy, companies must give customers what they want, not persuade them to purchase what the company now produces at lower cost."***[16]

A second problem with CA is it can encourage factories to produce excess inventory beyond what is really needed to fulfill customer orders. As noted earlier, pressure for high utilization of every machine and worker is a frequent cause of excess inventory. In addition, inventory accumulation can be driven by the counterintuitive effect it has on earnings.

Rather than being expensed on the income statement in the period incurred, the cost of inventory goes on the balance sheet as assets. Consequently, an inventory profit may be generated, which a business can use to smooth reported earnings even though it has essentially nothing to do with real income. However, if that inventory cannot be sold, the accounting eventually unwinds: Inventory on the balance sheet turns into depreciation expense on the income statement and an inventory loss results. Moreover, excess inventory creates competitive disadvantage because it hinders production and ties up capital that might otherwise be used to generate real income.

A third problem with CA concerns management priorities. In most companies, the dominant measurements are not bottom-line measurements, except perhaps at the executive level. Operating expense tends to be managed closely because it is well-known and under direct control. In contrast, revenue tends to be viewed as less controllable because it depends on markets and customers, which are not under direct control. Inventory tends to be a distant third in management priorities because reducing it has an adverse effect on reported income.

As explained earlier, however, inventory actually has a substantial impact on the ability of a business to compete. And although revenue is not under direct control, it is not bounded in the same way that inventory and operating expense are. That is, there are limits on how much inventory and operating expense can be reduced and still have a viable business. Conversely, there is no upper limit on the revenue that a business can attain and still be viable.

TA addresses all these problems via a different measurement approach: It does not use product costs but eliminates incentives for excess inventory, and it reverses typical management priorities.[17] TA is not, however, a substitute for conventional financial reporting because publicly traded companies must comply with generally accepted accounting principles (GAAP). Fortunately, TA can be readily reconciled with GAAP reporting even though TA is a different approach to management accounting.[18]

Figure 3-8 illustrates selected elements of TA. Decision support is on the left; actual results are on the right. Some of the terms abbreviated in this figure will probably be familiar, but keep in mind that their computations are usually different from CA.

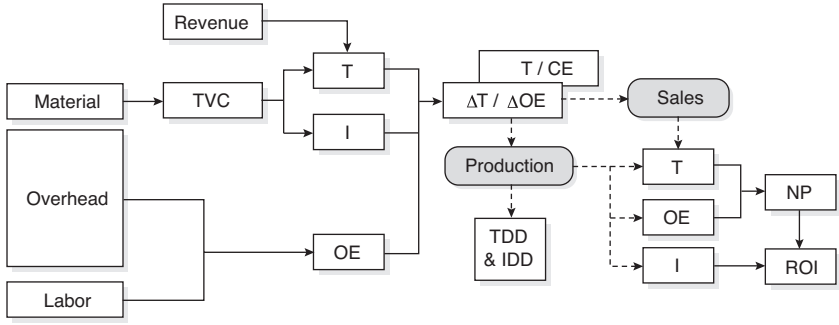

Figure 3-8 Throughput Accounting

TA begins with three financial measures:

- **Throughput (T)**—The rate at which money is generated through sales or interest. It is computed as revenue minus totally variable costs (TVC).
- **Inventory (I)**—All money invested in things intended for sale. It includes totally variable costs such as material, plus resources used in production such as land, machines, trucks, and computers. The more conventional term, Investment, is sometimes used instead of Inventory.
- **Operating Expense (OE)**—All money spent turning Investment into Throughput. It includes direct labor, rent, and labor, plus selling, general, and administrative (SG&A) costs.

T is maximized by selling goods or services with the largest difference between price and totally variable cost and by minimizing time between spending money to produce and receiving money from sales. Thus, T is determined by speed as well as magnitude, which is why the other TOC applications are designed to minimize delays.

Note that TA does not use labor costs to allocate OE. Furthermore, direct labor itself is not treated as a variable cost because enterprises do not adjust their workforce every time demand varies.

The financial measures are used to compute these performance measures:

- Net Profit: NP = T – OE
- Return on Investment: ROI = NP / I
- Productivity: P = T / OE
- Inventory Turns: i = T / I

An ideal decision increases T and decreases both I and OE. A good decision increases NP, ROI, Productivity, or Turns. Note that NP is net operating profit before interest and taxes.

Under TA, there are no product costs. Instead, there are constraint measures:

- Throughput per Constraint Unit: T/CU = (revenue – totally variable cost) / units
- Constraint Utilization: U = time spent producing / time available to produce

The way to maximize T is to maximize these constraint measures. Constraint utilization is important because each hour lost on the constraint is an hour lost by the entire factory or office. However, utilization of nonconstraints is not tracked because it encourages excess inventory.

Decisions based on T/CU include the following:

- Prioritizing use of the constraint (for example, choosing the best product mix)
- Deciding whether to increase the constraint's capacity via investment
- Selecting products to introduce or discontinue
- Pricing products based on the opportunity cost of using the constraint

Thus, for normal product decisions, T/CU is used to determine the mix that maximizes T. If producing less of one product in order to produce more of another product would increase T, for example, that's a good decision. But for major decisions that might shift the constraint or forfeit some T on current products, the following decision-support measure is better:[19]

- Change in Net Profit: $\Delta NP = \Delta T - \Delta OE$

The delta symbol (Δ) comes from mathematics and stands for "difference." It thus represents a comparison between alternatives.

Maximizing ΔNP ensures that major decisions improve profit across all products. And the following measure shows the impact of such investment decisions:

- Payback: $PB = \Delta NP / \Delta I$

To minimize unfavorable deviations from plans, these control measures should be minimized:

- Throughput Dollar Days: TDD = Selling price of late order × days late
- Inventory Dollar Days: IDD = Selling price of excess inventory × days unsold

TDD measures something that should have been done but was not: Ship orders on time. IDD measures something that should not have been done but was: Create unnecessary inventory.

The ideal value of these control measures is zero. The larger the value of either measure, the stronger the signal that corrective action is needed. Of course, if an enterprise uses a currency other than dollars, these measures can be recast as Throughput Value Days (TVD) and Inventory Value Days (IVD).[20]

In summary, TA is used to identify constraints, monitor performance, control production, and determine the impact of particular decisions. Yet the logic behind CA has been taught and practiced for so long that it can be hard to appreciate TA without an example.

Figure 3-9 compares CA to TA via a product mix decision. An actual decision could have hundreds of steps and thousands of products, which is one reason why redoing the analysis whenever the constraint moves is a problem. But this scenario consisting of just three products, with each product requiring the same three steps, is sufficient to illustrate key concepts. In this example, each product may require a different number of minutes per step, but the total time required by each product is the same. Furthermore, labor costs per minute are the same across all steps.

Product A has the highest price and lowest raw material cost per unit. Conversely, Product C has the lowest price and highest raw material cost per unit. Because the same workers will be used to produce any feasible product mix, the best mix would seem to be to produce as much of Product A as demanded, then B, then C. Following this priority, the factory will produce 100 units of A, 75 of B, and zero of C. Note that Step 2 limits enterprise production regardless of whether it's actually recognized as the constraint.

Operating expense includes rent, energy, and labor. When CA allocates operating expense to products based on their raw material costs, the resulting product costs confirm the expected priority: A has a lower product cost than B. Unfortunately, with this product mix, the enterprise generates a net loss. Because Product A appears to be profitable while B generates a loss, it's tempting to conclude that producing none of B would stop the loss. However, the operating expense covered by B would then have to be covered entirely by A, which would yield an even larger loss. Furthermore, if additional work were started in an effort to keep the workers at Steps 1 and 3 fully utilized, work-in-process inventory would grow. Therefore, by any measure, CA says this enterprise is unprofitable.

	Products				
	A	B	C	Have	Need
Demand	100	100	100		
Price	$105	$100	$95		
Raw Material	$45	$50	$55		
Step 1 Time	3	6	9	2,400	1,800 minutes
Step 2 Time	15	12	9	2,400	3,600 "
Step 3 Time	2	2	2	800	600 "
Total Time	20	20	20		

Cost Accounting	A	B	C	Total
Product Cost	$100	$111		
Mix	100	75	0	
Step 2 Used	1,500	900	0	2,400
Revenue	$10,500	$7,500	$0	$18,000
Raw Material	$4,500	$3,750	$0	$8,250
Gross Margin	$6,000	$3,750	$0	$9,750
Operating Expense	$5,455	$4,545	$0	$10,000
Net Profit	$545	-$795	$0	-$250

Throughput Accounting	A	B	C	Total
T / CU	$60	$50	$40	
T/CU / t	$4.00	$4.17	$4.44	
Mix	20	100	100	
Step 2 Used	300	1,200	900	2,400
Revenue	$2,100	$10,000	$9,500	$21,600
Raw Material (TVC)	$900	$5,000	$5,500	$11,400
Throughput (T)	$1,200	$5,000	$4,000	$10,200
Operating Expense (OE)				$10,000
Net Profit (NP)				$200

Figure 3-9 Cost Accounting versus Throughput Accounting

TA provides an entirely different perspective, however. TA ranks product profitability according to throughput on the constraint per minute (T/CU/t). And it does not allocate operating expense to products. Hence, Product A yields $4 per minute on the constraint, but B yields $4.17, and C yields $4.44. Therefore, TA says the priority should be to produce as much of C as capacity will allow, then B, then A. This is, of course, precisely the opposite priority seen previously. Because step 2 is the constraint, producing 100 units of C, 100 of B, and 20 of A is all that can be done.

With this product mix set via TA, the enterprise generates a net profit. Yet the only difference between CA and TA in this example is the product mix. All assumptions are precisely the same. Thus, CA does not optimize the enterprise. Indeed, this enterprise might not even survive if steered by CA.

Effective use of TA requires different information from CA, so new report formats must be implemented. For example, a TA earnings statement shows T, I, and OE relative to the constraint, while conventional CA reports are oblivious to constraints. Furthermore, just as CA and TA rank product profitability differently, they may also rank customer profitability quite differently.

Several TA outcomes are noteworthy:

- Financial measures reverse management priorities from OE, T, I to T, I, OE.
- Performance measures are not distorted by cost allocations.
- Constraint measures eliminate conflict between local measures (machine or worker utilization) and global measures (enterprise performance).
- Control measures remove incentive to build excess inventory and establish incentive to deliver on time.

Hence, previous TOC applications turned push into pull. TA tells the enterprise what to pull.

Implications for Services

Despite their obvious differences, industrial and services enterprise all have constraints. Thus, if the standard TOC applications summarized in this chapter are so useful in industry, it's reasonable to wonder why they haven't seen wider adoption in services. To be fair, each TOC application is used somewhere in services—but not widely. There are many reasons.

First, TOC is concerned with inventory, but what constitutes inventory in a services business can be harder to pin down. Is it billable hours, laboratories, software, servers, databases, equipment, methodologies, libraries, templates, deliverables, skills, or reputation? And if any or all of these constitute inventory, how would reducing them optimize a services enterprise? Even if we call those items investments rather than inventory, the optimal level of services investment isn't really addressed by conventional TOC applications. This is because services investments can be highly intangible and reusable, while industrial inventories more often are not.

Second, services in general are less repeatable than industry, and Professional, Scientific, and Technical Services are the most customized of all. When services and the steps they require change often, finding the constraint may require more than a little detective work. And by the time the constraint is located, it may hop elsewhere in the business. What's more, an enterprise that deliberately unbalances its services capacity to prevent a floating constraint may be committing itself to services that can't keep pace with what clients really want. So is it possible that floating constraints are unavoidable when services are delivered on demand?

Third, many services markets are moving away from services as available to services on demand. Even services businesses that historically operated with an internal constraint more often face a market constraint as technology and competition make alternatives plentiful and flexible. Where you once had to pick up your dry cleaning on Thursday, whether you wanted it then or not, now you can get it delivered in an hour. And cleaners are happy to accommodate you because they can charge you a premium price for expedited service that uses no more time on the constraint than regular service. Moreover, by capturing your request for service on demand, the provider prevents you from taking your business elsewhere or foregoing service entirely.

Finally, the degrees of freedom in delivering services can be greater than in manufacturing—particularly when the services depend on creativity. Resources with different skills and experiences can sometimes deliver services that are virtually indistinguishable when given the right tools and coaching. Resources aren't completely interchangeable, of course, but any sizable services contract can probably be configured in more than a hundred different ways to make the most of special talents and compensate for issues that arise unexpectedly.

The standard TOC applications have been used by service providers where their services sufficiently resemble industry. TOC hasn't seen widespread adoption in services overall, however, because innovations take time, and innovation adoption takes even more time. TOC has taken more than two decades to reach its current usage level in industry, and even today few enterprises use all the TOC applications. TOC in services simply hasn't reached the tipping point to wide adoption yet. Making it more widely applicable in services is, however, the purpose of this book.

Table 3-1 compares TOC for Goods (TOC_G) to TOC for Services (TOC_S). It thus provides a preview of the chapters ahead. Standard applications can be applied in essentially the same manner by any enterprise, while unique applications always must be tailored to a specific enterprise and its customers. Though every TOC_G application has a counterpart in TOC_S, every TOC_S application differs in significant ways.

Table 3-1 TOC for Goods Versus TOC for Services.

	TOC Application	TOC for Goods (TOC_G)	TOC for Services (TOC_S)
Standard	Drum-Buffer-Rope	DBR_G—manage manufacturing process	DBR_S—manage service process
	Replenishment	R_G—manage inventory	R_S—manage skilled resources
	Critical Chain	CC_G—manage multiple projects around strategic resource	CC_S—manage multiple projects using R_S
	Throughput Accounting	TA_G—manage finances around products	TA_S—manage finances around deliverables and service levels
Unique	Marketing and Sales	$M\&S_G$—use Buy-in to make customers want products because they solve core problems	$M\&S_S$—use Buy-in to make clients want services because they solve core problems
	Strategy and Change	$S\&C_G$—use Buy-in to bring about strategic change in goods producer	$S\&C_S$—use Buy-in to bring about strategic change in service provider *as well as clients*
	Implementation and Technology	$I\&T_G$—change goods producer's business rules, and then use technology to follow new rules	$I\&T_S$—change service provider's *or clients'* business rules, and then use technology to follow new rules

Summary

Drum-Buffer-Rope (DBR), Replenishment (R), Critical Chain (CC), and Throughput Accounting (TA) are four standard applications comprising Theory of Constraints (TOC) in industry. Each application solves a specific set of problems that otherwise prevent optimization of an enterprise.

DBR is the application for operations. By identifying one machine type or worker type as the constraint and scheduling production around it, DBR enables a factory to produce more than it can when every machine and worker are fully utilized.

R is the application for distribution. By distributing goods from a central warehouse in response to daily sales, rather than shipping large batches of goods to retail locations where sales are most variable, R increases sales at the same time it decreases inventory. Furthermore, R can be used within factories to much the same effect.

CC is the application for engineering. By managing resource contention as well as the precedence between tasks, CC eliminates bad multitasking, a significant productivity drain. Furthermore, by adopting estimating methods, work rules, and management procedures that focus on the constraint, projects managed with CC are not only shorter, but are more likely to finish on time and within budget.

TA is the application for finance and accounting. By focusing on Throughput before Operating Expense, TA reverses typical management priorities. By providing measures that show how to globally rather than locally optimize the enterprise, TA steers the enterprise toward its goal: to make money now and in the future (or produce goal units).

The common theme running through all TOC applications is constraint management. Because constraints are what keep an enterprise from reaching its goal, global optimization of enterprises has to address constraints.

Endnotes

1. Eliyahu Goldratt and Jeff Cox, *The Goal: A Process of Ongoing Improvement*, North River Press, 2nd Revised Edition, 1992.
2. H. William Dettmer, *Goldratt's Theory of Constraints: A Systems Approach to Continuous Improvement*, ASQ Quality Press, 1997.
3. Lisa Scheinkopf, *Thinking for a Change: Putting TOC Thinking Processes to Use*, St. Lucie Press, 1999.
4. Michael Spenser, "Theory of Constraints in a Service Application: The Swine Graphics Case," *International Journal of Production Research*, 38:5, 2000, pp. 1101–1108.

5. Lloyd Taylor, Brian Moersch, and Geralyn McClure Franklin, "Applying the Theory of Constraints to a Public Safety Hiring Process," *International Public Management Association for Human Resources*, Fall 2003.
6. R. Kershaw, "Using TOC to Cure Healthcare Problems," *Management Accounting Quarterly*, 2000, pp. 22–28.
7. David Anderson, *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*, Prentice-Hall, 2004.
8. Kelvyn Youngman, "A Guide to Implementing the Theory of Constraints (TOC)," www.dbrmfg.co.nz, 2005.
9. Eliyahu Goldratt, Eli Schragenheim, and Carol Ptak, *Necessary But Not Sufficient*, North River Press, 2000.
10. Eli Schragenheim and H. William Dettmer, *Manufacturing at Warp Speed: Optimizing Supply Chain Financial Performance*, St. Lucie Press, 2000, pp. 151–152.
11. Victoria Mabin and Steven Balderstone, *The World of the Theory of Constraints*, St. Lucie Press, 2000, pp. 11–12.
12. Eliyahu Goldratt, *It's Not Luck*, North River Press, 1994.
13. Eliyahu Goldratt, *Critical Chain*, North River Press, 1997.
14. Eliyahu Goldratt, *The Haystack Syndrome*, North River Press, 1990.
15. H. Thomas Johnson and Robert S. Kaplan, *Relevance Lost: The Rise and Fall of Management Accounting*, Harvard Business School Press, 1991.
16. H. Thomas Johnson, *Relevance Regained*, Free Press, 1992, pp. 149–151.
17. Thomas Corbett, *Throughput Accounting: TOC's Management Accounting System*, North River Press, 1998.
18. Debra Smith, *The Measurement Nightmare: How the Theory of Constraints Can Resolve Conflicting Strategies, Policies, and Measures*, St. Lucie Press, 2000.
19. Eli Schragenheim, "Throughput Based Decision Support," *TOC Review*, 2001.
20. Tim Sullivan, Richard Reid, and Brad Cartier, *TOC ICO Dictionary*, TOC International Certification Organization, 1st Edition (draft), August 2005.