11

Advanced Topics in Planning and Scheduling

Agile and Critical Chain

Chapter Outline

Chapter Objectives

After completing this chapter, you should be able to:

1. Understand why Agile Project Management was developed and its advantages in planning for certain types of projects.
2. Recognize the critical steps in the Agile process as well as its drawbacks.
3. Understand the key features of the Extreme Programming (XP) planning process for software projects.
4. Distinguish between critical path and critical chain project scheduling techniques.
5. Understand how critical chain methodology resolves project resource conflicts.
6. Apply critical chain project management to project portfolios

PROJECT MANAGEMENT BODY OF KNOWLEDGE CORE CONCEPTS COVERED IN THIS CHAPTER

1. Rolling Wave Planning Method Sequence Activities (PMBoK sec. 6.2.2.2)
2. Sequence Activities (PMBoK 6.3)
3. Estimate Activity Resources (PMBoK sec. 6.4)
4. Estimate Activity Durations (PMBoK sec. 6.5)
5. Develop Schedule (PMBoK sec. 6.6)
6. Develop Schedule (tools and techniques) (PMBoK sec. 6.6.2)
7. Critical Chain Method (PMBoK sec. 6.6.2.3)
8. Resource Optimization Techniques (PMBoK sec. 6.6.2.4)

PROJECT PROFILE

Developing Projects Through Kickstarter—Do Delivery Dates Mean Anything?

It is no longer the case that creative projects and their developers must convince corporate executives or venture capitalists to invest in their ideas. With the advent of the Kickstarter online platform, entrepreneurs can tap into a new source of funding through a concept known as crowdsourcing. The idea is simple: Set up a website, demonstrate the idea, and enlist thousands of enthusiastic supporters to donate money to your vision. To give an idea of just how successful the Kickstarter model has become, since its launch in April 2009, the site has already raised over $1 billion from hundreds of thousands of supporters to successfully fund some 57,000 projects. These projects have ranged from new electronic gadgets, to smartphone apps, to film projects, music recordings, video games, and other creative endeavors. Kickstarter is an all-or-nothing approach: If a project doesn't reach its funding goal, no one's credit card is charged. It seems that the Kickstarter model could be the future for successful "guerilla" project development, right?

Unfortunately, the track record of these projects is not so clear-cut. For example, in spite of the 57,000 successfully funded projects to date, estimates suggest that another 75,000 projects were unsuccessful. Of the "success stories," a closer examination raises some uncomfortable questions about them. For example, studies have suggested that for video games funded through Kickstarter between 2009 and 2012, just 37% were fully launched. Projects in the tech and design categories have seen similar levels of poor results: 75% of these projects experience significant delays in their launch. In fact, investors who pledge cash for a crowdfunded project had better be very patient: A CNNMoney examination of the top 50 most-funded projects on Kickstarter found that 84% missed their target delivery dates.

In examining the causes of these delays, a consistent pattern emerges. A team of ambitious but inexperienced creators will launch a project that they expect to attract a few hundred backers. Instead, the idea takes off, raising more—often a lot more—money than anticipated, and at the same time destroying the original production plans and timeline.

For example, consider Oculus Rift, a virtual reality headset designed by 20-year-old Palmer Luckey. He planned to make a few hundred headsets by hand. Then Kickstarter backers pre-ordered 7,500 units.

"In the first 24 hours, everyone is happy and slapping your hand," says Oculus CEO Brendan Iribe. "And 48 hours later, the reality sets in. There's a bit of fear: We're going to have to make all of these."

This unexpected success comes with serious strings attached; these developers have not thought their projects through to the delivery stage and certainly not for the volume of interest they attract. When plans get pushed, not all financial supporters are understanding. Oculus Rift was supposed to launch in November 2012—just two months after its Kickstarter campaign ended—but the shipping dates have been steadily eroding, as technical and production problems have slowed down delivery. Current plans are to have a consumer version of its virtual reality headset in stores by 2016.

"Kickstarter should alter their rules to include a section that limits postponing the delivery date," backer Martin F wrote on Oculus Rift's Kickstarter page. "If you postpone by 100% of the original delivery date, you are a scam."

The most common reasons that new projects did not meet their delivery dates include:

1. Manufacturing obstacles—creative originators of project ideas rarely give serious thought to the challenges of developing and delivering those projects, including the steps needed for production scheduling and quality management.
2. Shipping—although shipping sounds like a simple process, for large batches of promised deliveries worldwide, it can quickly become a nightmare. There are examples of companies that stopped all work and brought their entire staff, including programmers, to the loading dock just to pack their products for shipping.

(*continued*)

3. Volume—successful project developers are nearly always unprepared for the level of interest in their products and grossly underestimate the numbers they will have to create. As a result, their supply chain sourcing is not well planned, resulting is serious delays.
4. Apple's "curve ball"—Apple Corporation nearly single-handedly destroyed the production schedules of hundreds of new projects when they redesigned their "Lightning connector" for powering their products. These smaller entrepreneurs were so dependent on working off Apple's technology that any major shifts for the tech giant had serious repercussions for their delivery plans.
5. Changing scope—raising more money than expected sounds like a blessing but some firms used this extra funding as an excuse to drastically change the scope of the original project, opting for more expensive or expansive designs, technology, or materials. All these changes lead to serious delivery delays.
6. Certifications—any new device, such as a smartphone app, linked to the Apple iPhone has to receive the company's permission. Sometimes technologies must be approved for use by governmental agencies. These certifications all take time and delay the project.
7. Kickstarter's infrastructure—Kickstarter does not require that financial backers submit their addresses when they make pledges; all contact information has to be sought by the companies themselves at a later date. It is time-consuming and difficult to track down every investor and some have complained that the Kickstarter site functions more like a social network than an e-commerce Web site like eBay or Amazon.
8. Overseas logistics—a significant portion of Kickstarter's project investors are from other countries, making the delivery process (and costs associated with it) complicated. For example, once a Kickstarter campaign has closed, creators can't use Kickstarter's payment system to collect extra payments from their backers. That requires project developers to go back and solicit additional money to pay for the higher cost of overseas shipping.

Although Kickstarter offers fledgling entrepreneurs an opportunity to develop their projects without the oversight and financial commitment that comes from working for larger corporations, it also takes these creative people outside of their comfort zone when they are required to actually complete and deliver their project ideas within the promised timeframe. For many project developers, this has not been a problem; Kickstarter has proven to be an excellent source of funding and creative support. Kickstarter takes a 5% cut of each funded project, but it makes clear in its service terms that it isn't responsible for anything that happens after the cash changes hands. It won't get involved in disputes between users. As a result, for investors in these projects, the message can be summed up: Buyer beware.[1]

INTRODUCTION

Scheduling approaches that rely on CPM/PERT are generally accepted as standard operating procedure by most project-based organizations. Complications often occur, however, when we begin linking resource requirements or revised and changing customer needs to our developed schedules. As we will see in Chapter 12, the problem of constrained resources often reduces the flexibility we may have to create an optimal project schedule. Likewise, we noted earlier in the text that a critical goal of all projects is customer satisfaction with the finished project. Unfortunately, rigid project plans and schedules often do not allow the project team to maximize client satisfaction or make best use of limited resources. In recent years, however, two developments in project planning and scheduling have led to some important improvements in how we deliver projects: **Agile Project Management** and **Critical Chain Project Management (CCPM)**. Agile has become widely employed, particularly in the software and new product development industries, as a means for better linking project development with critical stakeholders, including customers and top management. Critical Chain, developed in the mid-1990s by Dr. Eli Goldratt, offers some important differences and advantages over more commonly used critical path methodologies. Lucent Technologies, Texas Instruments, Honeywell, and the Israeli aircraft industry are among a diverse group of major organizations that have found the underlying premises of CCPM intriguing enough to begin disseminating the process throughout their project operations.[2]

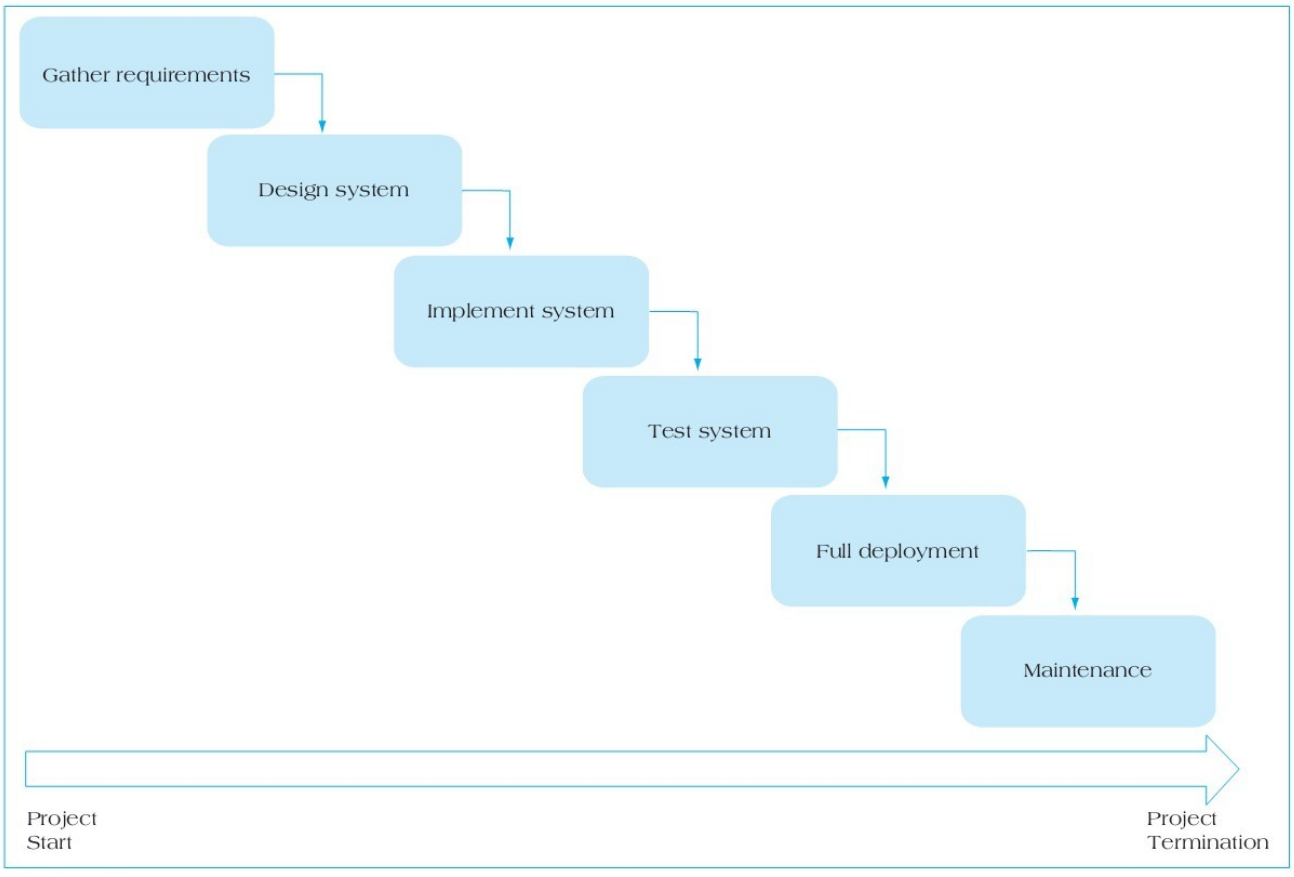
This chapter will explore in detail some of the important components of Agile project management and Critical Chain. We will see how, as supporters contend, these alternative scheduling mechanisms improve stakeholder satisfaction, promise to speed up project delivery, make better use of project resources, and more efficiently allocate and discipline the process of implementing projects. The Agile methodology represents a unique and very promising means to improve the way projects are planned and developed, using an iterative cycle with intriguing ideas like "Sprints" and "Scrums." We will also discuss a variant of the Agile

model used in software development, referred to as Extreme Programming (XP). A key feature of Agile and CCPM is that they represent both a cultural shift and a change in planning and scheduling processes. In practice, if either the Agile or CCPM methods are to be correctly applied, important technical and behavioral elements must be understood in relation to each other. The chapter will focus on these aspects of the process.

11.1 AGILE PROJECT MANAGEMENT

In recent years, a new project planning methodology has become increasingly important as organizations recognize that the traditional, highly structured approach to planning and managing projects may not be as effective for all types of projects. This realization is particularly true in the fields of information technology (IT) and new product development, where the end result may be impossible to fully visualize because of changing customer needs or consumer tastes. As a result, many organizations value flexibility in their project development practices, the ability to react quickly to opportunities or the need for changes mid-stream. **Agile Project Management (Agile PM)** reflects a new era in project planning that places a premium on flexibility and evolving customer requirements throughout the development process. Agile PM differs from traditional project management in a number of ways, but most particularly through recognizing that the old approach of carefully "planning the work and then working the plan" does not take into account the reality of many modern projects; that is, customer needs may evolve and change over the course of the project. Following the original plan that made sense when the project started would make no sense as the project progresses through the execution stage. Agile PM recognizes the importance of these evolving customer needs and allows for an incremental, iterative planning process—one that stays connected to clients across the project life cycle.

Agile PM offers an alternative to the traditional, waterfall planning process, which uses a linear, sequential life cycle model (see Figure 11.1). In the waterfall process, the assumptions that



guide project planning and development follow a logical series of steps. So, for example, in our waterfall model, each stage in this software development process occurs sequentially, following completion of the previous stage; first requirements are fully gathered, then the system is designed, implemented, and tested. Finally, it is fully deployed and subject to maintenance. In a **waterfall model**, each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In the waterfall model phases do not overlap. The waterfall project development process works well when:

- Requirements are very well understood and fixed at the outset of the project.
- Product definition is stable and not subject to changes.
- Technology is understood.
- Ample resources with required expertise are available freely.
- The project is of relatively short duration.

But what happens if requirements change in the middle of the project's development? Or the customer delivers a new set of "critical" features that have to be part of the final product? Or when a new technological innovation allows our team to streamline the software we are working on to make it more user-friendly? What if the initial assumptions or project scope were poorly communicated or a competitor beats us to the market with the identical product? Under these circumstances, the decision to agree to a fully articulated project scope, set in stone at the outset of the project, can lead to critical problems. With more and more projects supporting new product development initiatives that need the flexibility to change course midstream, the waterfall model can lead to a rigid process that will not deliver value to the customer, either because their needs are constantly changing or because they did a poor job of initially articulating exactly what the project was supposed to do. It is to address these critical issues that Agile PM was created.

What Is Unique About Agile PM?

Traditional project management establishes a fairly rigid methodology. As we remember from Chapter 1, the project life cycle suggests that conceptualization and planning occur at the start of the project. Conceptualization includes building a business case for the project (Why are we doing this? What do we wish to create? Can we make a profit or create value?) as well as identifying key project stakeholders. Planning focuses on creating the actual schedules and specifications for the project. In this traditional project management model, these critical activities are expected to happen early and be addressed comprehensively, but once completed they are considered done; in other words, it's now time to move onto the project execution. Underlying this traditional approach is the assumption that project members can identify risks (assume minimal uncertainty) and work their previously developed plans (assume maximum stability).

For many types of projects—short term, small scope, construction, event planning, and process improvement—these traditional project assumptions are not wrong. Projects with limited or narrow goals or those that will complete quickly generally do not suffer from significant problems of uncertainty. Their short timeframe minimizes environmental disruptions. Likewise, construction, current product upgrades, event planning, and other types of projects with clear goals and standard processes for completion can be carefully planned, have their costs accurately estimated, and schedules reasonably developed. Traditional project planning techniques do work well with certain classes of project or in certain situations. However, the more unpredictable the environment, including having to take into account changing customer tastes, the challenge of disruptive technologies, complex development processes, and long time frames, the greater the likelihood that the traditional project planning approach, which assumes stability and predictable development, will not succeed.

It is to address the problems with traditional project planning that innovative new techniques like Agile were first developed. Agile PM, usually referred to as **Scrum**, recognizes the mistake of assuming that once initial project conceptualization and planning are completed, the project can simply be executed to original specifications, and thus ensure the completed project will be a "success." For example, software projects are prone to constant change. Yet, when clients are expected to finalize all requirements and wait for an extended period before even viewing prototypes, the likelihood is extremely high of creating projects greeted by, "That's not what I wanted!" Creating a plan and then disengaging from the customer during its development may be a traditional

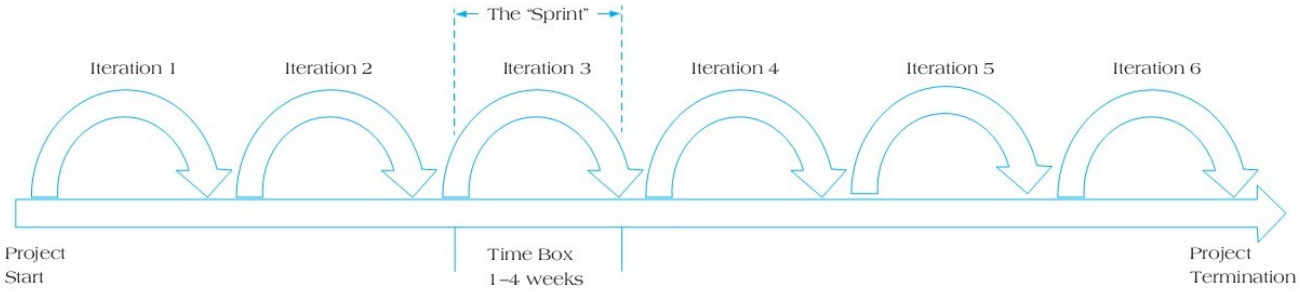


FIGURE 11.2 Scrum Process for Product Development

approach to project management but it is dangerous with technologically complex or uncertain projects. Agile PM is a flexible, iterative system designed for the challenge of managing projects in the midst of constant change and uncertainty, so it moves the planning function from its traditional up-front location in the project life cycle to occur throughout project development. In effect, Agile PM makes development a "rolling wave" process of continuous plan–execute–evaluate cycles across project development (see Figure 11.2). The goal of each wave in Scrum is to create incremental value, through steadily developing subfeatures or elements in the overall project. The length of these development iterations is kept deliberately short (from one to four weeks)—long enough to create some valuable addition to the project that customers can evaluate but short enough to remain in constant communication and respond to immediate requests or requirements modifications. Following each development cycle, a review meeting occurs in which project features are evaluated, changes agreed to, specifications modified, and next deliverables identified.

To illustrate with a software project example, Agile PM reduces the complexity and uncertainty of an inefficient and costly traditional approach by avoiding the common mistake of a months-long process of building requirements for the overall project, finishing the product and then testing it to find hundreds of bugs and unneeded or nonworking features. Instead, smaller but still usable subfeatures or portions of the software project would be completed one at a time, tested and verified, all in shorter time periods. In this case, any changes to the software are not hugely expensive in terms of either time or money.

Scrum originated in the quality work of Takeuchi and Nonaka,[3] who promoted a holistic method for new product development. In their model, they argued that development teams must work as an integrated unit to reach their goals. The traditional, sequential approach is sometimes referred to as "over the fence" because it illustrates a model where each functional group adds something to the product and then, when finished with their efforts, tosses the project "over the fence" to the next functional group that is expected to continue the development process. This method slows down new product development, leads to the failure to capture critical product features, encourages miscommunication and functional rivalries, and hugely increases the costs of fixing products late in their development cycle, when technical errors and feature misunderstandings can lead to failed projects and wasted money.

Instead of running a "relay race," Takeuchi and Nonaka encouraged project organizations to look to a different sport—rugby—in order to develop the right mindset for creating new products. The old approach, they argued, consisted of passing off the project development "baton" to the next group in the relay and moving toward the finish line with minimal interplay between development partners or the chance to work together in a coordinated, interdisciplinary way. Rugby, on the other hand, emphasizes adaptation, flexibility, and the coordinated efforts of multiple players, all working to the same purpose, but adjusting and modifying their efforts as the project proceeds. The whole process is performed by one cross-functional team across multiple overlapping phases, where the team "tries to go the distance as a unit, passing the ball back and forth."[4]

Tasks Versus Stories

One of the critical differences between Agile and traditional project planning relates to the nature of the roles that key members assume. In a traditional project planning process, the project developer's viewpoint is considered most important. Developers are looking at the project from the

inside perspective; that is, how long will development take? How many work packages and tasks are necessary to complete the project? Project developers are interested in tasks because they allow them to accurately and efficiently plan the work and build their cost estimates. The more detailed and specific the project is made, the easier it is to create these plans and estimates.

User stories are different and very important for understanding the real needs of the client (the "voice of the customer"). They are written by or for customers as a means for influencing the development of the functionality of the product. The more the customer can explain what they do, what they need, and how they can make use of the product to do their job better, the clearer the user story becomes and the better able the Scrum team will be to achieve these goals. Stories have value because they identify the actual work that needs to be done with the completed product or system. Once stories have been validated, they can then be decomposed into tasks. The key lies in recognizing the need to first listen to user stories to identify the specific value-added the project will provide. For example, a project that ignores the voice of the customer can be well-managed, brought in on time and under budget, but provide no real value to the customer because it was done without regard to first identifying critical user stories (what they need to do their job better).

Another Agile characteristic that demonstrates the importance of the voice of the customer is the emphasis on product **features**, as opposed to creating a detailed product WBS. We assume in an Agile environment that project scope characteristics are inevitably going to change over the course of the project's development. As a result, it makes little sense to invest heavily in time and effort to develop a sophisticated statement of scope for the overall product. Instead, Agile focuses on getting the product features right: the pieces of the product that deliver functional value to the customer. Listening for customers' stories and defining the project's critical deliverables in terms of features reinforces the critical nature of the ongoing, closely linked relationship the Scrum team must maintain with clients.

Key Terms in Agile PM[5]

Sprint—A Sprint is one iteration of the Agile planning and executing cycle. So, the Sprint represents the actual "work" being done on some component of the project and must be completed before the next Scrum meeting.

Scrum—In the sport of rugby, a "scrum" is the restarting of the game following a minor infraction. In Agile PM, Scrum refers to the development strategy agreed to by all key members of the project. Scrum meetings involve assessing the current status of the project, evaluating the results of the previous Sprint, and setting the goals and time-box for the next iteration.

Time-box—A time-box is the length of any particular sprint and is fixed in advance, during the Scrum meeting. As we mentioned, time-boxes typically vary between one and four weeks in length.

User stories—A short explanation, in the everyday language of the end user that captures what they do or what they need from the project under development. The goal of the user story is to gain their perspective on what a correctly developed product will do for them.

Scrum Master—The Scrum Master is the person on the project team responsible for moving the project forward between iterations, removing impediments, or resolving differences of opinion between the major stakeholders. The Scrum Master *does not* have to be the project manager but does have a formal role in enforcing the rules of the Scrum process, including chairing important meetings. Scrum Masters are focused solely on the Agile project development process and do not play a role in people management.

Sprint Backlog—A Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the Sprint Goal. The Sprint Backlog is a forecast by the development team about what functionality will be in the next time-box increment and the work needed to complete that functionality. The team controls the Sprint Backlog.

Burndown Chart—The Sprint Burndown Chart shows remaining work in the Sprint backlog. Updated every day and displayed for all Scrum members to see, it provides a quick reference of the Sprint progress.

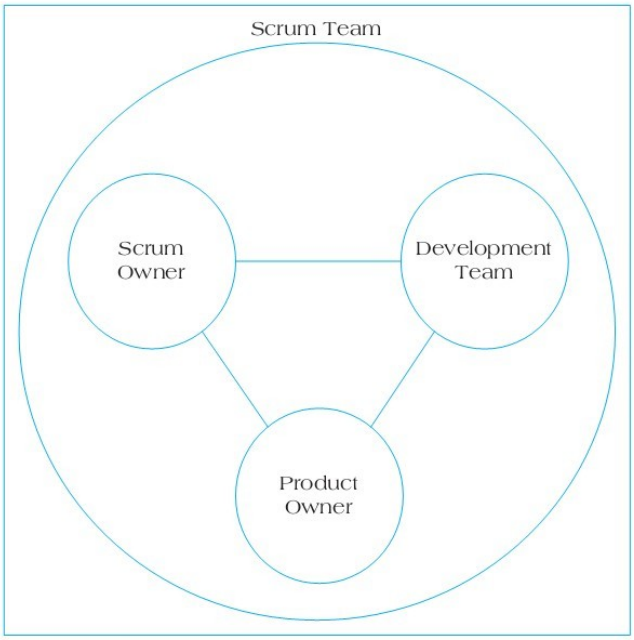


FIGURE 11.3 Members of the Scrum Team

Product Owner—The person representing the stakeholders and serving as the "voice of the customer." The product owner may be a member of the project organization but must take the "outside," user's viewpoint in representing customer needs. The product owner creates **user stories** that identify their specific needs for the product.

Development team—The organizational unit responsible for delivering the product at the end of each iteration (Sprint). Typically, the development team is cross-functional and self-organizing; that is, they collectively determine the best way to achieve their goals.

Product Backlog—The Product Backlog is a prioritized list of everything that might be needed in the completed product and is the source of requirements for any changes to be made to the product. The Product Backlog is never "final"; it evolves as the product and the business setting in which it will be used evolve. It constantly changes to identify what the product needs to be appropriate, competitive, and useful. The product owner controls the Product Backlog.

Work backlog—The evolving, prioritized queue of business and technical functionality that needs to be developed into a system.

Steps in Agile[6]

The Agile process follows a series of steps that allow the methodology to combine the flexibility needed to respond to customer needs with a formal process that creates a logical sequence in employing Agile planning. The Scrum process involves a set of meetings that manage the project development process through (1) Sprint Planning, (2) Daily Scrums, (3) the Development Work, (4) Sprint Review, and (5) Sprint Retrospective (see Figure 11.4). During the Sprint, three guidelines shape the process:

1. No changes are made that would endanger or modify the Sprint goal. Once goals for the Sprint are agreed to, they are not to be altered in the middle of the Sprint.
2. Quality goals do not decrease. During the Sprint, the team cannot modify the goals or sacrifice quality standards that were first agreed to.
3. Scope may be clarified and renegotiated between the product owner and the development team as more is learned. As the team discovers technical problems or opportunities during the Sprint, they are passed to the product owner for consideration on whether or not to modify the project scope.

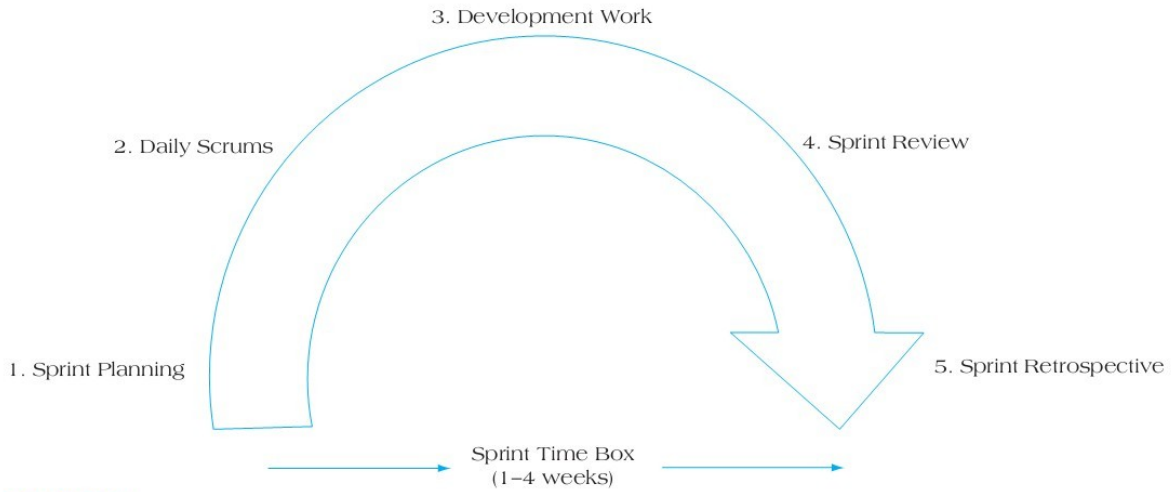


FIGURE 11.4 Stages in a Sprint

Sprint Planning

The work to be performed in the Sprint is identified during the Sprint Planning session. This plan is created by the collaborative work of the entire Scrum team. Sprint Planning is time-boxed to a maximum of one full day (eight hours) for a one-month Sprint. For shorter Sprints, less time is required for Sprint Planning. The Scrum Master ensures that the event takes place and that all members of the Scrum team understand its purpose. The Sprint Planning session introduces the Sprint Backlog to the development team and coordinates their efforts to achieve the Backlog items.

Sprint Planning answers the following questions:

- What can be delivered in the increment (time-box) resulting from the upcoming Sprint?
- How will the work needed to deliver the increment be achieved?

Daily Scrums

The Daily Scrum is a short (15 minutes) event that allows the development team an opportunity to synchronize their activities and create a plan for the next 24-hour time window. During the meeting, members of the development team explain what they accomplished in the past 24 hours to meet the Sprint goal, what they intend to work on during the current day, and identify any problems that might prevent the development team from completing the next Sprint goal. Daily Scrum sessions are for information purposes; they are only intended to keep team members in the communications loop and identify any positive or negative trends affecting the project. Daily Scrums also include reference to the Burndown Chart, detailing the latest information on the status of Backlog items completed ("burned down") since the last Scrum meeting.

The Development Work

The Development Work is the time when the actual work of the project is being done during the Sprint. These are the set of goals that must be achieved during the Sprint and are represented on the Burndown Chart as either in progress or completed. Development Work must be heavily coordinated between Scrum team members to ensure that no efforts are being wasted or work is being done on non-Sprint items. Figure 11.5 shows an example of a Burndown Chart for a Sprint that assumes the following details:

- Sprint duration – 15 days
- Team size – 5 members
- Hours/day – 8
- Total capacity – 600 hours

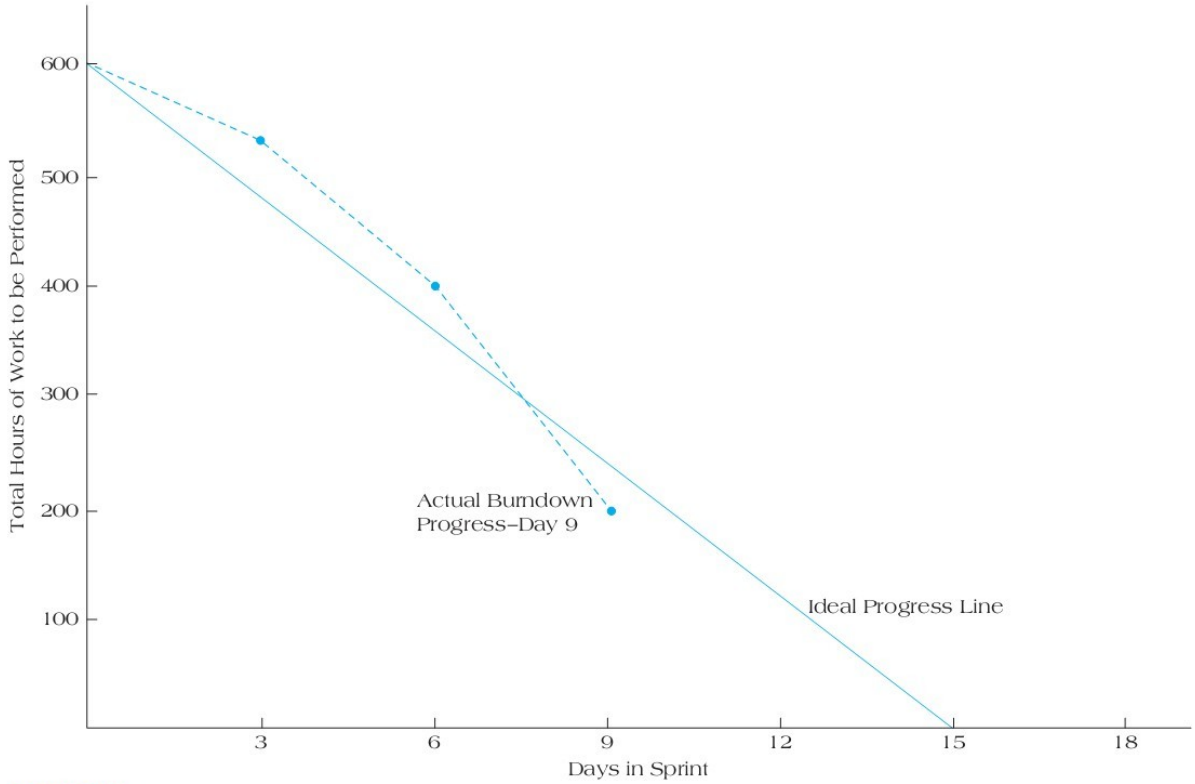


FIGURE 11.5 Sample Burndown Chart for Day 9 of a Sprint

Sprint Reviews

A Sprint Review is held at the end of the Sprint to inspect the completed increment (the Sprint Backlog) and make changes to the Product Backlog if needed. During the Sprint Review, the Scrum team and other key stakeholders work closely to verify what was done on the Sprint. Based on these results and any subsequent changes to the Product Backlog, the Scrum team now plans the next things to be done in order to add value, including product features that need completing or modifying. Sprint Reviews are informal meetings; they are not status meetings, and the presentation of the completed Sprint Backlog is only to encourage team feedback and encourage collaboration. The result of the Sprint Review meeting is a modified Product Backlog and an idea of the Sprint Backlog items that will be addressed in the next Sprint. During the Sprint Review, the activities to be addressed include the following:[7]

- The product owner explains what Product Backlog items have been completed and what has not yet been completed.
- The development team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved.
- The development team demonstrates the work that it has finished and answers questions about the latest Sprint.
- The product owner discusses the Product Backlog as it stands. He projects likely completion dates based on progress to date (if needed).
- The entire group collaborates on what to do next, so that the Sprint Review provides useful input to subsequent Sprint Planning.
- Review how the marketplace or potential use of the product might have changed the next most valuable thing to complete.
- Review the timeline, budget, potential capabilities, and marketplace for the next anticipated release of the product.

Sprint Retrospective

The Sprint Retrospective is a meeting that is held to evaluate how the previous Sprint went; what worked, what didn't work, and where potential improvements can be made to the Sprint process. A valuable Sprint Retrospective should also include an action plan for identifying and implementing improvements to the process. The Scrum Master works with the Scrum team to constantly improve communications and identify improvements that will be introduced during the next Sprint. In this way, each Sprint is not only about getting the work (the Product Backlog) done, it is also used to generate enthusiasm for the next Sprint while creating a more efficient and productive team.

Problems with Agile

Agile offers many advantages to project planners and developers, particularly within classes of project for which the goals of Agile are most relevant, such as IT project management or new product development. However, some disadvantages to the Agile methodology have to be considered as well. These disadvantages include:[8]

1. Active user involvement and close collaboration of the Scrum team are critical throughout the development cycle. This requirement is very time-demanding of all parties involved and requires users to commitment to the process from start to finish.
2. Evolving requirements can lead to the potential for scope creep throughout the development, as new options or changes during Sprints can result in a never-ending series of requested changes by the user.
3. Because of emerging requirements and the flexibility to make changes midstream, it is harder to predict at the beginning of the project what the end product will actually resemble. This can make it hard to make the business case for the project during the conceptualization phase or negotiate fixed-price contracts with customers or vendors.
4. Agile requirements are kept to a minimum, which can lead to confusion about the final outcomes. With the flexibility to clarify or change requirements throughout the project development, there is less information available for team members and users about project features and how they should work.
5. Testing is integrated throughout the lifecycle, which adds to the cost of the project because the services of technical personnel such as testers are needed during the entire project, not just at the end.
6. Frequent delivery of project features (the Sprint Backlog) throughout the project's incremental delivery schedule means that testing and signing-off on project features is nearly continuous. This puts a burden on product owners to be ready and active when a new set of features emerges from the latest Sprint cycle.
7. If it is misapplied to projects that operate under high predictability or a structured development process, the requirements of Agile for frequent rescoping or user input can be an expensive approach without delivering benefits.

BOX 11.1

Project Management Research in Brief

Does Agile Work?

The principles of Agile have been in existence for over 20 years, have been introduced and used in numerous organizations to streamline project development, and have been generally applauded as a sound planning and executing philosophy. Nevertheless, while anecdotal evidence suggests that Agile does lead to shorter development times and better, more customer-friendly outcomes, there is little empirical research that supports the claims that Agile PM does, in fact, work better than traditional project planning methods for some classes of project. A recent study of over 1,000 projects from a variety of settings—information technology (IT), software, construction, manufacturing, health care, and financial services—found that the "degree of Agile" in the company's project planning significantly affected project success. Projects that used higher levels of planning across the entire life cycle (the Scrum/Sprint iteration process) developed projects that performed better on meeting budget and schedule targets, as well as experienced higher levels of customer satisfaction.[9]

11.2 EXTREME PROGRAMMING (XP)

Extreme Programming (XP) is a more aggressive form of Scrum and is a software development methodology intended to improve software quality and responsiveness to changing customer requirements.[10] Originally developed by programmer Kent Beck for Chrysler Corporation, XP is deceptively simple in its core principles, which include an emphasis on keeping the programming code simple, reviewing it frequently, testing early and often, and working normal business hours. XP also adopts Agile's emphasis on the importance of user stories as a means for understanding their real needs, not a programmer's interpretation of a rigid scope statement. XP takes its name from the idea that innovative and beneficial elements of software engineering practices are taken to "extreme" levels with this approach.

Two of the guiding features of XP are the process of **refactoring** and **pair programming**. In order to speed software development, functional testing of all requirements is done before coding begins and automated testing of the code is performed continuously throughout the project. Refactoring is the continuous process of streamlining the design and improving code; not waiting until final testing to edit and fix code. Another controversial feature of XP is the philosophy of pair programming. In XP, all code is written in collaboration between pairs of programmers, who work side-by-side on the same machine during coding. Pair programming can help programmers resolve issues and clarify interpretation of user stories that drive the requirements. XP also requires constant communication between customers and the developer team. In fact, it has been suggested that project teams should never number more than 12 developers working in pairs. Other elements of Extreme Programming include avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. As originator Kent Beck notes:[11]

> "The basic advantage of XP is that the whole process is visible and accountable. The developers will make concrete commitments about what they will accomplish, show concrete progress in the form of deployable software, and when a milestone is reached they will describe exactly what they did and how and why that differed from the plan. This allows business-oriented people to make their own business commitments with confidence, to take advantage of opportunities as they arise, and eliminate dead-ends quickly and cheaply."

Agile PM (and XP) have grown out of the need to combine the discipline of a project management methodology with the needs of modern enterprises to respond quickly to opportunities, promote an internal operating environment of communication and collaboration, and remain connected and committed to customers. For many types of projects, Agile PM offers a means to streamline project development processes while improving bottom line results. By decreasing the costs of rework from misunderstood or changing customer requirements, Agile PM has been demonstrated to save project organizations money while improving customer relations and encouraging functional groups within the organization to work together in a cooperative manner. All in all, the Agile planning philosophy offers numerous advantages for organizations developing new products rapidly and cost-effectively.

11.3 THE THEORY OF CONSTRAINTS AND CRITICAL CHAIN PROJECT SCHEDULING

In practice, the network schedules we constructed in the previous two chapters, using PERT and probabilistic time estimates, are extremely resource dependent. That is, the accuracy of these estimates and our project schedules are sensitive to resource availability—critical project resources must be available to the degree they are needed at precisely the right time in order for the schedule to work as it is intended. One result of using "early-start" schedules is to make project managers very aware of the importance of protecting their schedule slack throughout the project. The more we can conserve this slack, the better "buffer" we maintain against any unforeseen problems or resource insufficiency later in the project. Thus project managers are often locked into a defensive

mode, preparing for problems, while they carefully monitor resource availability and guard their project slack time. The concept of theory of constraints as it is applied to Critical Chain Project Management represents an alternative method for managing slack time and more efficiently employing project resources.

Theory of Constraints

Goldratt originally developed the theory of constraints (TOC), first described in his book *The Goal* (1984), for applications within the production environment.[12] Among the more important points this author raised was the idea that, typically, the majority of poor effects within business operations stem from a very small number of causes; that is, when traced back to their origins, many of the problems we deal with are the result of a few core problems. The main idea behind TOC is the notion that any "system must have a constraint. Otherwise, its output would increase without bound, or go to zero."[13] The key lies in identifying the most central constraint within the system. Five distinct steps make up the primary message behind TOC methodology (see Figure 11.6):

1. ***Identify the system constraint.*** First, an intensive search must be made to uncover the principal constraint, the root cause, that limits the output of any system. It is important to not get bogged down in identifying numerous secondary causes or "little problems."
2. ***Exploit the system constraint.*** Once the constraint is identified, a strategy for focusing and viewing all activities in terms of this constraint is necessary. For example, if the constraint within a software development firm is having only one advanced application programmer, the sequence of all project work to be done by the programmer has to be first scheduled across the organization's entire portfolio of active projects.
3. ***Subordinate everything else to the system constraint.*** Make resource commitment or scheduling decisions after handling the needs of the root constraint. Using the above example, once the "critical resource constraint" of one programmer has been identified and the programmer's time has been scheduled across multiple projects, the rest of the project activities can be scheduled.
4. ***Elevate the system constraint.*** The first three steps acknowledge that the system constraint limits an organization's operations. According to Goldratt, the fourth step addresses improvement of the system by *elevating the constraint,* or seeking to solve the constraint problem by eliminating the bottleneck effect. In our software-programming example, this may mean hiring an additional advanced applications programmer. For many project-based examples, "elevating the system constraint" may be as simple as acquiring additional resources at opportune times.

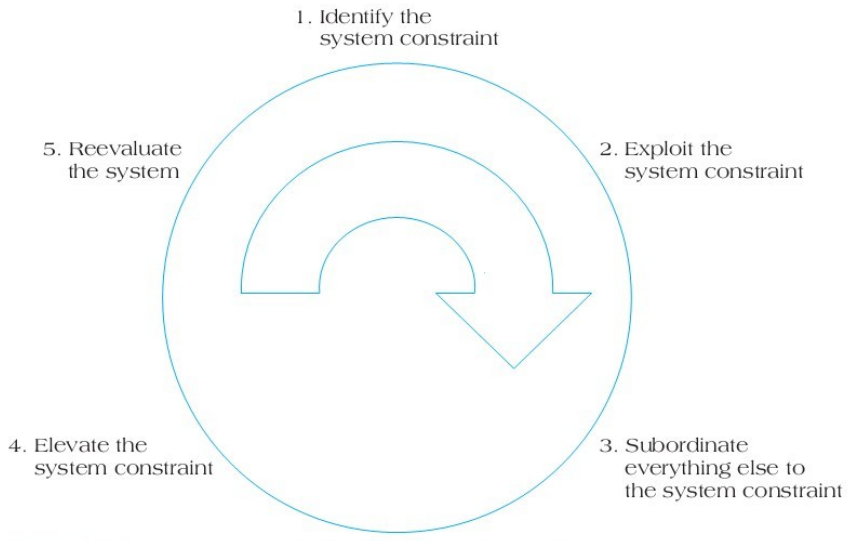


FIGURE 11.6 Five Key Steps in Theory of Constraints Methodology

5. ***Determine if a new constraint has been uncovered, and then repeat the process.*** Clearly, the removal of the key system constraint will lead to positive advantages for a time. Since there is *always* a system constraint, however, removing one constraint is only likely to identify a new source of constraint for the operation. TOC argues for the need to always prepare for the next potential problem before it becomes too serious, so this final step is really only one step in a continuous improvement cycle.

When examining a project schedule from the perspective of TOC methodology, we focus on the key system constraint, that is, the one root cause from which all other scheduling problems evolve. The system constraint for projects is initially thought to be the *critical path.* Remember, the critical path is defined as the earliest possible time on the activity network it can take to complete a project. If activities on the critical path are delayed, the effect is to cause delays in the overall project. Critical path is determined by the series of activities whose durations define the longest path through the network and therefore identify the project's earliest possible completion. Goldratt notes that all scheduling and resource problems associated with projects typically occur due to problems with trying to maintain the critical path, and hence its oft-made identification as the chief system constraint.[14]

11.4 THE CRITICAL CHAIN SOLUTION TO PROJECT SCHEDULING

Goldratt's solution to the variables involved in project scheduling involves the aggregation, or collectivizing, of all project risk in the form of uncertain duration estimates and completion times. The aggregation of risk is a well-known phenomenon in the insurance business.[15] The **central limit theorem** states that if a number of probability distributions are summed, the variance of the sum equals the sum of the variances of individual distributions. This formula is given, where there are n independent distributions of equal variance V, as:

$$V_{\Sigma} = n \times V$$

where V_{Σ} is the variance of the sum.

The standard deviation σ can be used as a surrogate for risk, and since $\sigma^2 = V$, we find:

$$\sigma_{\Sigma} = (n)^{1/2} \times \sigma$$

where σ_{Σ} is the standard deviation of the sum. Therefore:

$$\sigma_{\Sigma} < n \times \sigma$$

Mathematically, the above formula illustrates the point that aggregating risks leads to a reduction in overall risks.

This same principle of aggregation of risks can be applied in a slightly different manner to the **critical chain** methodology. We have used the term *safety* or *project buffer* to refer to the contingency reserve for individual activities that project managers like to maintain. When we aggregate risk, this reserve is dramatically reduced so that all activity durations are realistic but challenging. That is, rather than establish duration estimates based on a 90% likelihood of successful completion, all activity durations are estimated at the 50% level. The provision for contingency, in the form of project safety, is removed from the individual activities and applied at the project level. Because of the aggregation concept, this total buffer is smaller than the sum of the individual project activity buffers. Thus project duration is reduced.

Apple Computer Corporation's recent success story with its iPad tablet illustrates some of the advantages to be found in aggregating risks. Apple made a conscious decision with the iPad to subcontract most of the components of the product to a variety of suppliers. The company determined that to engineer the entire product would have been a complex and risky alternative. Instead, it contracted with a number of suppliers who had produced proven technology. The decision to combine these product components from other sources, rather than manufacture them in-house, led to a much faster development cycle and greatly increased profitability.[16]

Two fundamental questions to be answered at this point are: Exactly how much is the project's duration reduced? How much aggregated buffer is sufficient? Goldratt and his adherents do not advocate the removal of all project buffer, but merely the reapplication of that buffer to a project level (as shown in Figure 11.7). The determination of the appropriate amount of buffer to be maintained can be derived in one of two ways: (1) a "rule of thumb" approach that Goldratt suggests, namely, retain 50% of total project buffer; and (2) a more mathematically derived model suggested by Newbold (1998):[17]

$$\text{Buffer} = \sigma = [((w_1 - a_1)/2)^2 + ((w_2 - a_2)/2)^2 + \ldots + ((w_i - a_i)/2)^2]^{1/2}$$

where w_i is the worst-case duration and a_i is the average duration for each task that provides part of the aggregated buffer value. The presumed standard deviation would be $(w_i - a_i)/2$. Suppose, for example, that the project team sought a buffer that is 2 standard deviations long. The formula for calculating an appropriate buffer length is:

$$\text{Buffer} = 2 \times \sigma = 2 \times [((w_1 - a_1)/2)^2 + ((w_2 - a_2)/2)^2 + \ldots + ((w_i - a_i)/2)^2]^{1/2}$$

Let us assume, for example, that we have three tasks linked together, each of 20 days in length. Thus, the worst case (w_i) for these durations is the original 20 days. Further, by aggregating the buffer based on a 50% solution, our a_i value is 10 days for each activity. We can solve for the appropriate buffer size (two standard deviations) by:

$$\begin{aligned}\text{Buffer} &= \sqrt{((20_1 - 10_1)^2 + (20_2 - 10_2)^2 + (20_3 - 10_3)^2)} \\ &= \sqrt{300}, \text{ or } 17.32 \text{ days}\end{aligned}$$

Visually, we can understand the application of CCPM in three distinct phases. First, all relevant project tasks are laid in a simplified precedence diagram (shown on line 1 in Figure 11.7), with anticipated durations specified. Remember that the original duration estimates have most likely been based on high probability of completion estimates and therefore require a reexamination based on a more realistic appraisal of their "true" duration. The second step consists of shrinking these duration estimates to the 50% likelihood level. All individual task safety, or buffer, has been aggregated and now is given as the project-level buffer.

At this stage, the overall length of the project has not changed because the individual task buffer is simply aggregated and added to the end of the project schedule. However, line 3 illustrates the final step in the reconfiguration, the point where the project buffer shrinks by some identifiable amount. Using the rule of thumb of 50% shrinkage, we end up with a project

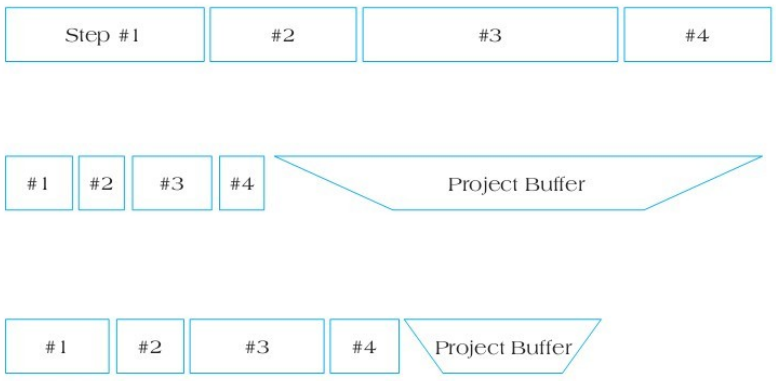


FIGURE 11.7 **Reduction in Project Duration After Aggregation**

Source: L. P. Leach. (1999). "Critical chain project management improves project performance," *Project Management Journal*, *30*(2), 39–51, figure on page 44. Copyright © 1999 by Project Management Institute Publications. Copyright and all rights reserved. Material from this publication has been reproduced with the permission of PMI.

TABLE 11.1 Critical Chain Activities Time Reductions

Activity	Original Estimated Duration	Duration Based on 50% Probability
A	10 days	5 days
B	6 days	2 days
C	14 days	7 days
D	2 days	1 day
E	8 days	3 days
Total	40 days	18 days

schedule that is still significantly shorter than the original. This modified, shortened schedule includes some minor slack for each activity. As a result, CCPM leads to shortened project schedules.

Suppose that a project activity network diagram yielded the initial values given in Table 11.1. Note that the modified network shortens the overall project duration by 22 days, from the original 40 to 18. Because all risk is now aggregated at the project level, there are a total of 22 days of potential slack in the schedule resulting from shrinking activity estimates at each project step. A CCPM-modified project schedule would reapply 11 days of the acquired schedule shrinkage to serve as overall project buffer. Therefore, the new project schedule will anticipate a duration estimated to require 29 days to completion.

What are the implications of this reapplication of project slack to the aggregated level? First, all due dates for individual activities and subactivities have been eliminated. Milestones are not used in the CCPM activity network. The only firm commitment remains to the project delivery date, not to the completion of individual tasks. Project team members are encouraged to make realistic estimates and continually communicate their expectations. Clearly, in order for CCPM to work, a corporate culture that supports a policy of "no blame" is vital. Remember, the nature of requiring 50% likelihood estimates for individual activity durations implies that workers are just as likely to *miss* a commitment date as to achieve it. Under a culture that routinely punishes late performance, workers will quickly reacquire the habits that had once protected them—inflated estimates, wasting safety, and so forth.

A second implication may be more significant, particularly when dealing with external subcontractors. Because individual activity dates have been eliminated and milestones are scrapped, it becomes problematic to effectively schedule subcontractor deliveries. When subcontractors agree to furnish materials for the project, they routinely operate according to milestone (calendar) delivery dates. CCPM, with its philosophy that deemphasizes target dates for individual tasks, creates a complicated environment for scheduling necessary supplier or subcontractor deliveries. Writers on CCPM suggest that one method for alleviating this concern is to work with contractors to negotiate the early completion and delivery of components needed for critical activities.[18]

Developing the Critical Chain Activity Network

Recall from earlier chapters that with traditional CPM/PERT networks, individual activity slack is an artifact of the overall network. Activity start time is usually dictated by resource availability. For example, although an activity could start as early as May 15, we may put it off for three days because the individual responsible for its completion is not available until that date. In this way, float is used as a resource-leveling device.

With CCPM, resource leveling is not required because resources are leveled within the project in the process of identifying the critical chain. For scheduling, therefore, CCPM advocates putting off all noncritical activities as late as possible, while providing each noncritical path in the network with its own buffer (see Figure 11.8). These noncritical buffers are referred to as *feeder buffers* because they are placed where noncritical paths feed into the critical path. As Figure 11.8 demonstrates, a portion of the critical path and one of the noncritical feeder paths join just past the point of activity C. Feeding buffer duration is calculated similarly to the process used to create the overall project buffer, attached to the end of the critical chain.

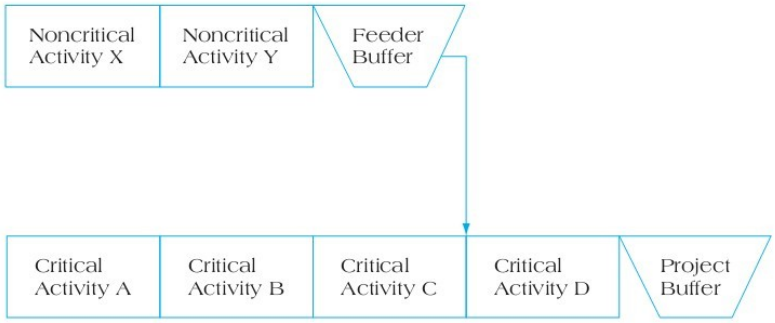


FIGURE 11.8 CCPM Employing Feeder Buffers

Note: Feeder buffers are intended to prevent delays on critical activities.

To understand how the logic of the critical chain is constructed, note that the first steps lie in making some important adjustments to traditional scheduling approaches, such as:

1. Adjusting expected activity durations to reflect a 50% probability of completion on time (shrinking the schedule)
2. Changing from an early-start process to a late-finish approach
3. Factoring in the effects of resource contention if necessary

Figures 11.9a, b, and c present a simplified series of examples that follow these steps. Figure 11.9a shows a standard activity network based on a PERT approach. A total of five activities are identified (A, B, C, D, and E) along two separate paths feeding into activity E at the project's conclusion. All activities are scheduled to begin as early as possible (early start) and are based on a standard method for estimating durations. Table 11.2 lists these expected durations.

Figure 11.9a demonstrates an expected overall project duration of 90 days, based on the longest set of linked activities (path A – B – E). The second path, C – D – E, has an overall duration of 60 days and hence, has 30 days of slack built into it. In order to adjust this network, the first step involves changing to a late-start schedule. Second, CCPM challenges the original activity duration estimates and substitutes ones based on the mean point of the distribution. The modified activity network makes the assumption of shrinking these estimates by 50%. Therefore, the new network has an overall duration of 45 days, rather than the original 90-day estimate (Figure 11.9b).

The next step in the conversion to a critical chain schedule involves the inclusion of project and feeder buffers for all network paths. Recall that these buffers are calculated based on applying 50% of the overall schedule savings. The feeder buffer for the path C – D is calculated as (.50)(10 + 5), or 7.5 days. The project buffer, found from the values for path A – B – E, is calculated as (.50)(5 + 25 + 15), or 22.5 days. Hence, once buffers are added to the modified activity network, the original PERT chart showing duration of 90 days with 30 days of slack, the new critical chain network has an overall duration of 67.5 days, or a savings of 22.5 days (Figure 11.9c). Through three steps, therefore, we move from an early-start to a late-start schedule, identify the critical path (sequence of longest linked activities), and then apply feeder and project buffers. The result is a modified project schedule, which, even with buffers inserted, significantly reduces scheduled completion time for the project.[19]

TABLE 11.2 Activity Durations

Activity	Duration
A	10 days
B	50 days
C	20 days
D	10 days
E	30 days

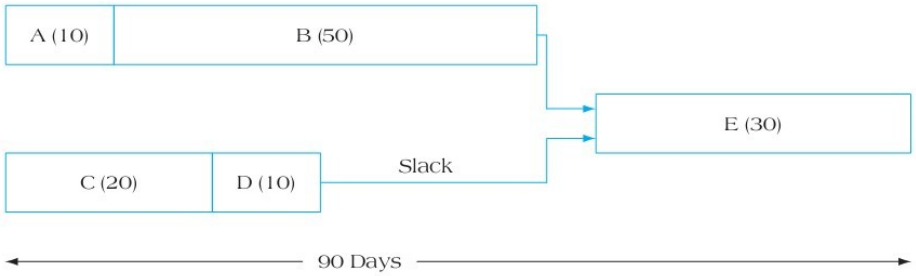


FIGURE 11.9a Project Schedule Using Early Start

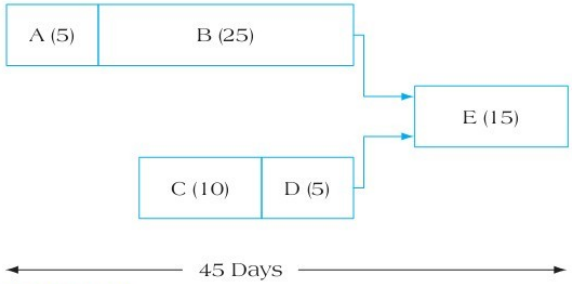


FIGURE 11.9b Reduced Schedule Using Late Start

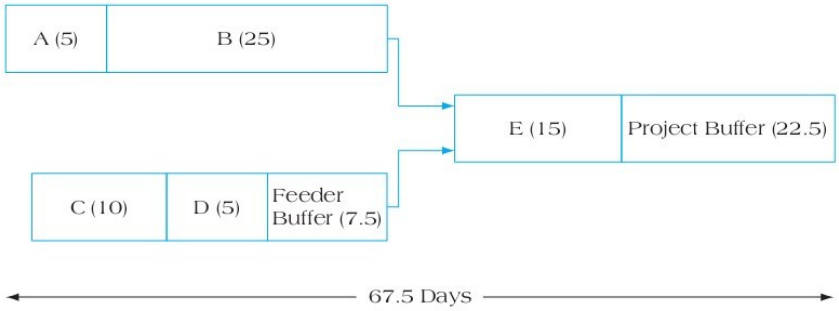


FIGURE 11.9c Critical Chain Schedule with Buffers Added

Critical Chain Solutions Versus Critical Path Solutions

So what is the real difference between the critical path method and Critical Chain Project Management? Critical chain is usually not the same path as the critical path within an activity network. The critical path depends only on task dependency, that is, the linkage of tasks with their predecessors. In this process, activity slack is discovered after the fact; once the network is laid out and the critical path identified, all other paths and activities may contain some level of slack. On the other hand, the critical chain usually jumps task dependency links. Again, this effect occurs because critical chain requires that all resource leveling be done before the critical chain can be identified, not afterward as in the case of PERT and CPM networks.

To illustrate this distinction, consider the differences when the activity network in Figure 11.10a is compared with the modified solution in Figure 11.10b. Figure 11.10a shows a simplified PERT network that identifies three paths. The central path is the critical path. The difficulty occurs when we require the same resource (Bob) to complete activities that are scheduled simultaneously. Clearly, Bob cannot perform the three tasks at the same time without significantly lengthening the overall critical path. The alternative, shown in Figure 11.10b, is to first resource-level the activities that Bob must perform. The project's schedule must take into account the resource conflict and demonstrate a new network logic that allows the project to proceed.

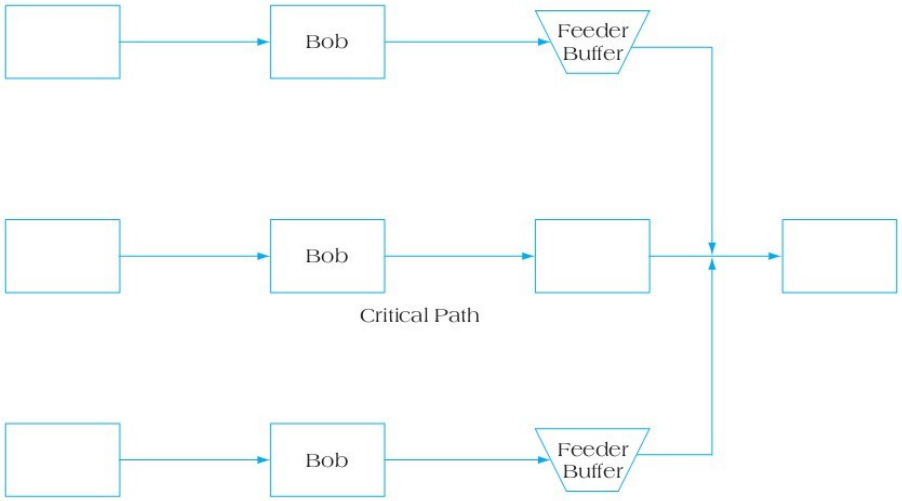


FIGURE 11.10a Critical Path Network with Resource Conflicts

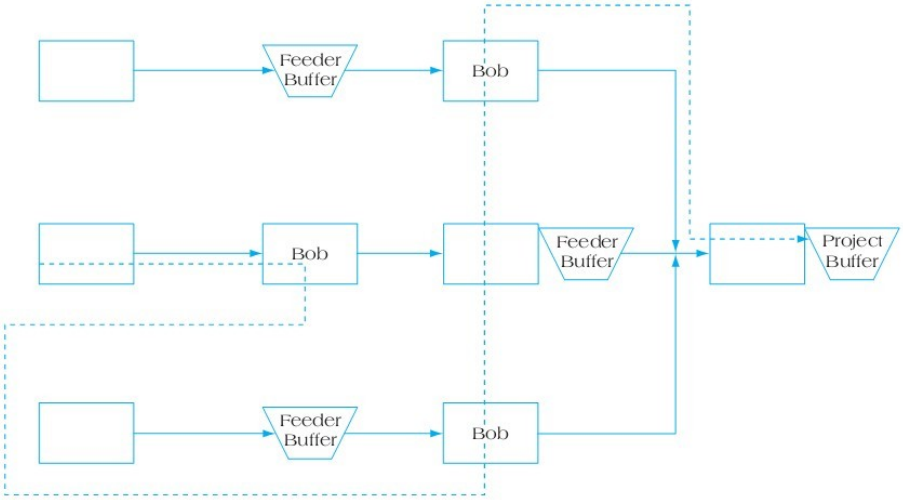


FIGURE 11.10b The Critical Chain Solution

Note: The critical chain is shown as a dashed line.

Bob, our resource constraint (Figure 11.10b), forces the schedule to be redrawn in order to reflect his work assignments. Note that with the critical chain schedule (shown with the dashed line), Bob first completes his task on the central path. The other two paths require Bob as well, and so he is first assigned to the task on the lower path and he then accounts for his final assignment, along the top path. Also note how the various feeder buffers must be redrawn in the new critical chain schedule. Because Bob's work on the first task is the predecessor for his subsequent activities, the feeder buffers on the top and bottom schedules are moved forward, or earlier, in the network to account for his resource availability (if he is delayed). Hence, because Bob is the critical resource in the network, it is imperative to first level him across the tasks he is responsible for and then redraw the network to create a new critical chain, which is distinct from the original critical path. Once the critical chain is identified, feeder buffers are added to support the critical activities while providing a margin of safety for the noncritical paths.

PROJECT PROFILE

Eli Lilly Pharmaceuticals and Its Commitment to Critical Chain Project Management

Eli Lilly Corporation is one of the giants in the pharmaceutical industry, but in the drug-manufacturing industry, size is no guarantee of future success. All pharmaceutical firms in the United States are facing increasing pressure from a variety of sources: (1) the federal government, which has just enacted elements of "Obamacare" with strict guidelines for drug cost control; (2) the loss of patents as key drugs become generic; and (3) the need to maintain leadership in a highly competitive industry. Lilly is beginning to feel this sting personally; starting in 2011, several of its top-selling drugs went off patent, leaving the company scrambling to bring new drugs into the marketplace quickly. Unfortunately, their "late-stage" pipeline is thin; there are few drugs waiting in the wings to be commercialized.

In its efforts to stay out in front, Lilly has announced a series of strategic moves. First, the firm has instituted a cost-cutting initiative across the organization in hopes of trimming more than $1 billion from operations. Second, Lilly has reorganized into four divisions in order to streamline and consolidate operations to become more market-driven and responsive. Finally, the firm has announced the formation of a Development Center of Excellence in R&D, to be sited at corporate headquarters in Indianapolis, Indiana. The Center will be responsible for accelerating the completion of late-stage trials and release of new drugs. What does Lilly see as being key to the success of its Center of Excellence? One important element is the widespread use of Critical Chain Project Management (CCPM).

Lilly has been championing CCPM in its R&D units since 2007 and is committed to instituting the process throughout its entire R&D organization. The company's support for CCPM is based on the results of hard evidence: "It has now been implemented on 40 of our new product pipeline and our projects are 100 percent on time delivery compared to about 60 percent for the other 60 percent of the [drugs] in the more traditional development programs,"[20] according to Steven Paul, President of Lilly Research Labs.

Lilly has found that CCPM gives the company multiple advantages, starting with re-creating a cooperative internal environment based on shared commitment of various departments to the drug development process. Further, CCPM offers a method for maximizing the efficiency of the firm's resources, avoiding common bottlenecks in the development cycle, and moving drugs through the trial stages much more rapidly. Finally, it encourages an internal atmosphere of authenticity in estimating, scheduling, and controlling projects.

The move to CCPM did not come easily. Some managers have noted that it requires a different mind-set on the part of employees, who have to see their projects from an "organizational" point of view rather than from a strictly departmental perspective. Nevertheless, Lilly's public commitment to CCPM has paid off and continues to serve as a catalyst for the company's competitive success.[21]

Noel Hendrickson/Blend Images/Alamy

FIGURE 11.11 Drug Discovery Research Lab

11.5 CRITICAL CHAIN SOLUTIONS TO RESOURCE CONFLICTS

Suppose that after laying out the revised schedule (refer back to Figure 11.9c), we discover a resource contention point. Let us assume that activities B and D require the same person, resulting in an overloaded resource. How would we resolve the difficulty? Because the start dates of all activities are pushed off as late as possible, the steps to take are as follows:

1. The preceding task for activity D is activity C. Therefore, the first step lies in assigning a start-as-late-as-possible constraint to activity C.
2. To remove the resource conflict, work backward from the end of the project, eliminating the sources of conflict.

Figure 11.12 presents an MS Project file that illustrates the steps in adjusting the critical chain schedule to remove resource conflicts. Note that the original figure (Figure 11.9c) highlights a standard problem when developing a typical early-start schedule, namely, the need to evaluate the schedule against possible resource overload. Suppose, for example, that the Gantt chart (Figure 11.12) indicates a resource conflict in the form of Joe, who is assigned both activities B and D during the week of March 6. Since this person cannot perform both activities simultaneously, we must reconfigure the schedule to allow for this constraint.

Figure 11.13 shows the next step in the process of resolving the resource conflict. While maintaining a late-start format, activity D is pushed back to occur after activity B, thereby allowing

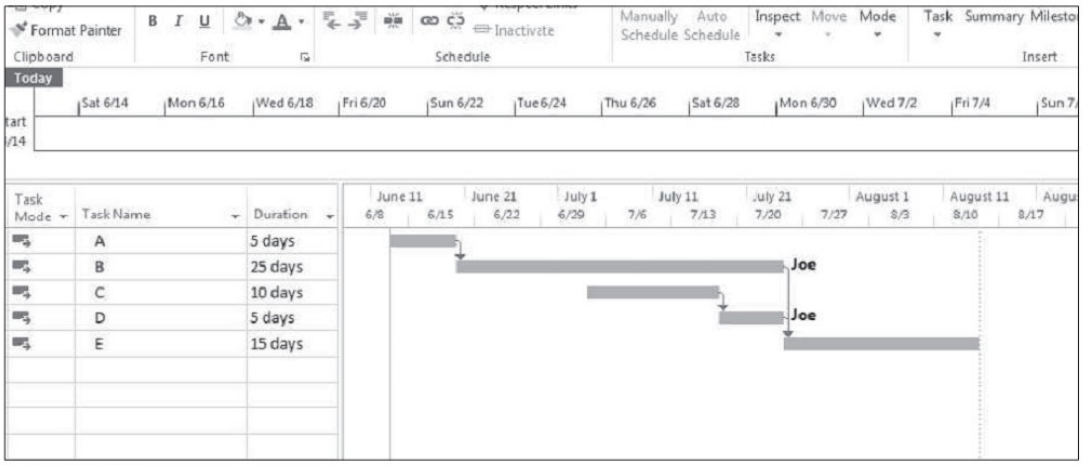


FIGURE 11.12 Scheduling Using Late Start for Project Activities

Source: MS Project 2013, Microsoft Corporation.

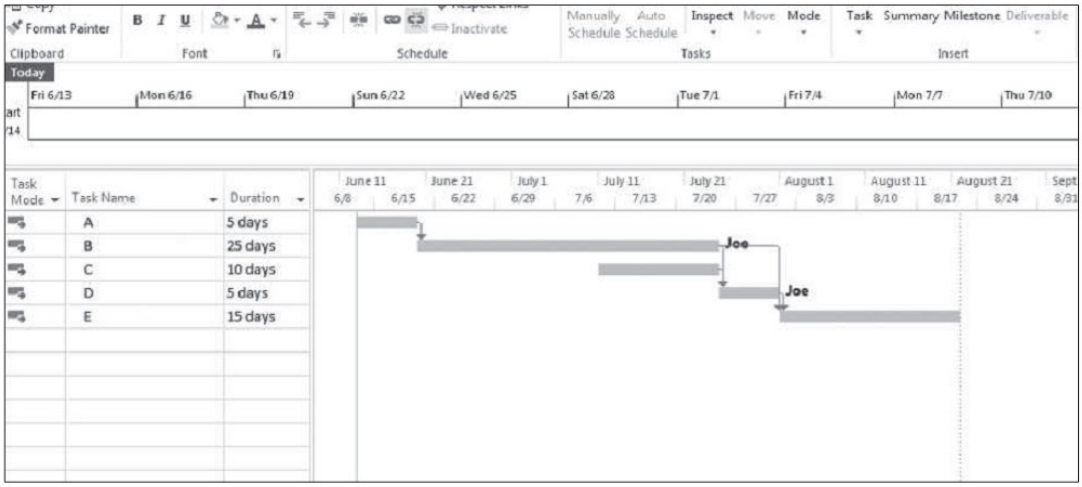


FIGURE 11.13 Reconfiguring the Schedule to Resolve Resource Conflicts

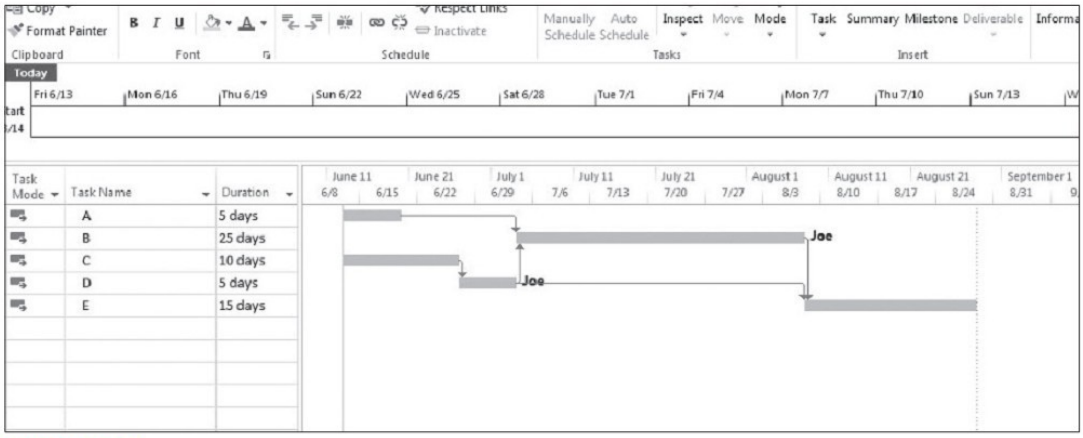


FIGURE 11.14 **Alternative Solution to Resource Conflict Problem**

Source: MS Project 2013, Microsoft Corporation.

Joe to first perform B before moving to his next assignment. The total schedule delay amounts to approximately one week with the reconfigured schedule.

Alternatively, this resource conflict problem can be rescheduled according to Figure 11.14, in which activities C and D are moved forward in the network. This alternative solution does add additional time to the network path, moving the projected completion date to the second week in April. When choosing the most viable solution to resource conflict issues, you want the option that minimizes total network schedule disruption. In the examples shown, it might be preferable to adopt the schedule shown in Figure 11.13 because it addresses the resource conflict and offers a reconfigured schedule that loses only one week overall.

11.6 CRITICAL CHAIN PROJECT PORTFOLIO MANAGEMENT

Critical Chain Project Management can also be applied to managing a firm's portfolio of projects. Basic TOC logic can be applied to the portfolio of company projects to identify the key systemwide constraint. Recall that in the single-project example, the key constraint is found to be the critical chain. At the organizationwide level, the chief constraint is commonly seen as the company's resource capacity. In balancing the portfolio of projects in process, we must first evaluate the company's chief resource constraints to determine available capacity. The resource constraint may be a person or department; it may be a companywide operating policy, or even a physical resource. In a production capacity, Goldratt has used the term **drum** in reference to a systemwide constraint, because this limiting resource becomes the drum that sets the beat for the rest of the firm's throughput.[22]

In order to apply CCPM to a multiproject environment, we must first identify the current portfolio of projects. Next, the chief resource constraint, or drum, is identified and, following TOC methodology, that system constraint is exploited. With project portfolio scheduling, this step usually consists of pulling projects forward in time because the drum schedule determines the subsequent sequencing of the firm's project portfolio. If the drum resource is early, some projects can be pulled forward to take advantage of the early start. If the drum is late, projects may need to be pushed off into the future. We also need to employ buffers in portfolio scheduling, much as we did for feeder paths and overall project buffering in individual project cases. The term **capacity constraint buffer (CCB)** refers to a safety margin separating different projects scheduled to use the same resource. Applying a CCB prior to sequencing to the next project ensures that the critical resource is protected. For example, if Julia is the quality assessment expert and must inspect all beta software projects prior to their release for full development, we need to apply a CCB between her transition from one project to the next. Finally, we can also use **drum buffers** in portfolio scheduling. Drum buffers are extra safety

applied to a project immediately before the use of the constrained resource to ensure that the

resource will not be starved for work. In effect, they ensure that the drum resource (our constraint) has input to work on when it is needed in the project.[23]

The formal steps necessary to apply CCPM to multiple project portfolios include:[24]

1. Identify the company resource constraint or the drum, the driving force behind multiple project schedules. Determine which resource constraint most directly affects the performance of the overall system or which is typically in short supply and most often requires overtime. Such physical evidence is the best indicator of the company's central constraint.
2. Exploit the resource constraint by—
   a. Preparing a critical chain schedule for each project independently.
   b. Determining the priority among the projects for access to the drum, or constraining resource.
   c. Creating the multiproject resource constraint, or drum, schedule. The resource demands for each project are collected and conflicts are resolved based on priority and the desire to maximize project development performance.
3. Subordinate the individual project schedules by—
   a. Scheduling each project to start based on the drum schedule.
   b. Designating the critical chain as the chain from the first use of the constraining resource to the end of the project.
   c. Inserting capacity constraint buffers (CCBs) between the individual project schedules, ahead of the scheduled use of the constraint resource. This action protects the drum schedule by ensuring the input is ready for it.
   d. Resolving any conflicts if the creation of CCBs adversely affects the drum schedule.
   e. Inserting drum buffers in each project to ensure that the constraint resource will not be starved for work. The buffers should be sited immediately before the use of the constraint resource in the project.
4. Elevate the capacity of the constraint resource; that is, increase the drum capacity for future iterations of the cycle.
5. Go back to step 2 and reiterate the sequence, improving operating flow and resource constraint levels each time.

As an example, consider Figure 11.15. We have identified a drum resource constraint, suggesting that the resource supply is not sufficient to accommodate all three projects (A, B, and C) that are queued to be completed. This point is illustrated by the dashed line running horizontally across the figure. One option, of course, is to drop the project with the lowest priority, in essence allowing the drum resource to dictate the number of projects that can be accomplished. Alternatively, we can consider methods for exploiting the system constraint through the use of capacity constraint buffers to accomplish all three projects, on their priority basis. Figure 11.15 shows the nature of the problem, with project A having the highest priority, B the next highest, and C the lowest priority. Resources exist to handle only two projects simultaneously, but the resources are not needed continuously, as the figure shows. As a result, the resource constraint problem really becomes one of scheduling, similar to the single-project case.

Once we have identified the resource constraint and prioritized the projects for access to the drum resource, we can reschedule the projects in a manner similar to that shown in Figure 11.16.[25] The problem is one of constrained capacity, so the task consists of pushing the additional project C off until such time as it can be included in the drum schedule. A capacity constraint buffer (CCB) is placed in front of the start date to begin work on project C. This buffer ensures that the critical resource is available when needed by the next project in the pipeline and defines the start date for the new project.

This same procedure can be used as we add a fourth, fifth, or sixth project to the portfolio. Each project is constrained by access to the drum resource and must, therefore, be scheduled to take into consideration the system constraint. By so doing, we are able to create a master project schedule that employs Goldratt's theory of constraints philosophy within a multiproject environment.

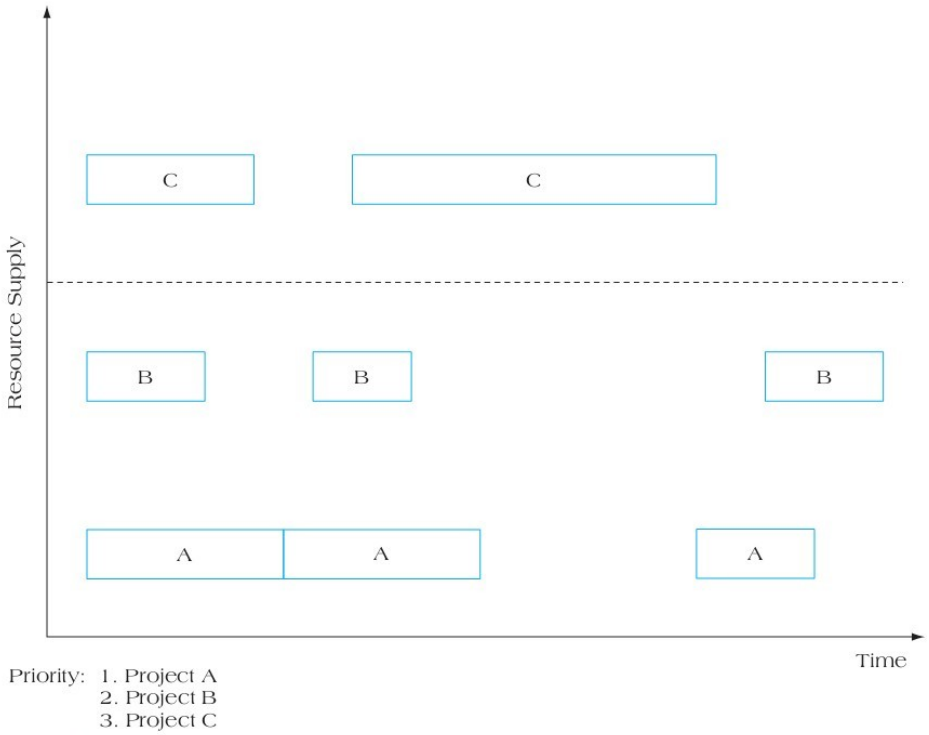


FIGURE 11.15 Three Projects Stacked for Access to a Drum Resource

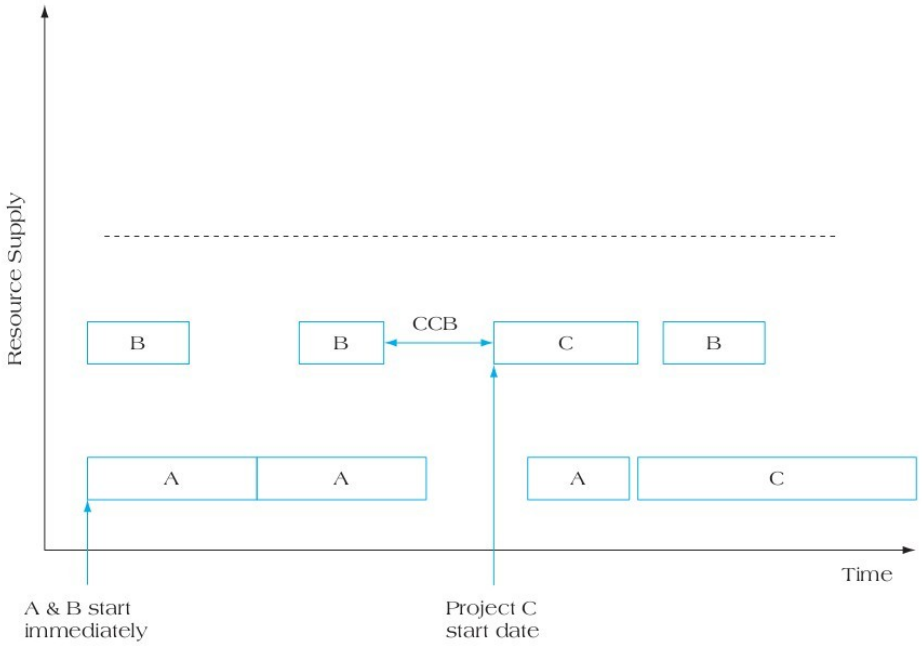


FIGURE 11.16 Applying CCBs to Drum Schedules

BOX 11.2

Project Management Research in Brief

Advantages of Critical Chain Scheduling

Does CCPM really work? Although a number of recent books and articles have appeared championing the methodology, little empirical evidence exists to date to either confirm or disconfirm the viability of the critical chain approach to scheduling. Evidence tends to be primarily anecdotal in nature, as CCPM advocates point to a number of firms that have realized significant savings in time and positive attitudinal changes on the part of project team members following the adoption of critical chain scheduling.

A recent study by Budd and Cooper[26] sought to test the efficacy of CCPM against traditional critical path scheduling in a simulation environment. Using three long projects and more than 1,000 iterations with both a critical chain and a critical path schedule, the authors projected completion times for the projects under study and determined that total activity durations for the critical chain schedules were shorter than durations using the critical path method. For their simulation models, the long projects under a CPM schedule were projected to take from 291 to 312 days to completion, with a mean finish time of 293 days. Critical chain projects were projected to take from 164 to 181 days, with a mean value of 170 days to completion. In fact, in multiple iterations involving different length projects, critical chain scheduling reduced the mean duration time to complete projects anywhere from 18% to 42%. The only caveat the authors noted was their inability to reflect the negative effects of multitasking on either schedule. Nevertheless, their findings offer some evidence in support of critical chain project management as a viable alternative to critical path scheduling.

Additional research evidence is also suggesting that CCPM does have a positive impact on project outcomes. In IT project management, reported results suggest that successfully adopting CCPM shows reductions in project durations of about 25%, increased throughput (the number of projects finished per unit of time) of 25%, and the number of projects completed on time rose to 90%. Finally, a compilation of recent results from different project settings offers some encouraging evidence (see Table 11.3).[27]

TABLE 11.3 Company Project Performance Improvements Using Critical Chain Project Management

CCPM Implementation	Before	After
New Product Development for Home Appliances (Hamilton Beach/Proctor-Silex)	34 new products per year. 74% of projects on time.	Increased to 52 new products in first year and to 70 + in second year. 88% of projects on time.
Telecommunications Network Design and Installation (eircom, Ireland)	On-time delivery less than 75%. Average cycle time of 70 days.	Increased on-time delivery to 98 + %. Average cycle time dropped to 30 days.
Helicopter Manufacturing and Maintenance (Erickson Air-Crane)	Only 33% of projects completed on time.	Projects completed on time increased to 83%.
Oil & Gas Platform Design & Manufacturing (LeTourneau Technologies, Inc.)	Design engineering took 15 months. Production engineering took 9 months. Fabrication and assembly took 8 months.	Design engineering takes 9 months. Production engineering takes 5 months. Fabrication and assembly takes 5 months with 22% improvement in labor productivity.
High Tech Medical Development (Medtronic Europe)	Device projects took 18 months on average and were unpredictable.	Development cycle time reduced to 9 months. On-time delivery increased to 90%.
Transformer Repair and Overhaul (ABB, Halle)	42 projects completed January–December 2007. On-time delivery of 68%.	54 projects completed January–December 2008. On-time delivery of 83%.

11.7 CRITIQUES OF CCPM

Critical Chain Project Management is not without its critics. Several arguments against the process include the following charges and perceived weaknesses in the methodology:

1. Lack of project milestones make coordinated scheduling, particularly with external suppliers, highly problematic. Critics contend that the lack of in-process project milestones adversely affects the ability to coordinate schedule dates with suppliers that provide the external delivery of critical components.[28]
2. The "newness" of CCPM is a point refuted by some who see the technique as either ill-suited to many types of projects or simply a reconceptualization of well-understood scheduling methodologies (such as PERT), provided special care has been taken to resource-level the network.[29]
3. Although it may be true that CCPM brings increased discipline to project scheduling, efficient methods for the application of this technique to a firm's portfolio of projects are unclear. The method seems to offer benefits on a project-by-project basis, but its usefulness at the program level has not been proven.[30] Also, because CCPM argues for dedicated resources, in a multiproject environment where resources are shared, it is impossible to avoid multitasking, which diminishes the power of CCPM.
4. Evidence of success with CCPM is still almost exclusively anecdotal and based on single-case studies. Debating the merits and pitfalls of CCPM has remained largely an intellectual exercise among academics and writers of project management theory. With the exception of Budd and Cooper's modeling work, no large-scale empirical research exists to either confirm or disconfirm its efficacy.
5. A recent review of CCPM contended that although it does offer a number of valuable concepts, it is not a complete solution to current project management scheduling needs. The authors contended that organizations should be extremely careful in excluding conventional project management scheduling processes to adopt CCPM as a sole method for planning and scheduling activities.[31]
6. Critics also charge that Goldratt's evaluation of duration estimation is overly negative and critical, suggesting that his contention that project personnel routinely add huge levels of activity duration estimation "padding" is exaggerated.
7. Finally, there is a concern that Goldratt seriously underestimates the difficulties associated with achieving the type of corporatewide cultural changes necessary to successfully implement CCPM. In particular, while activity estimate padding may be problematic, it is not clear that team members will be willing to abandon safety at the request of the project manager as long as they perceive the possibility of sanctions for missing deadlines.[32]

Successful implementation and use of CCPM is predicated first on making a commitment to critically examining and changing the culture of project organizations in which many of the problems identified in this chapter are apparent. Truth-in-scheduling, avoiding the student syndrome, transferring project safety to the control of the project manager—these are all examples of the types of actions that bespeak a healthy, authentic culture. Gaining "buy-in" from organizational members for this type of scheduling process is vital to the success of such new and innovative techniques that can dramatically improve time to market and customer satisfaction.[33]

Summary

1. **Understand why Agile Project Management was developed and its advantages in planning for certain types of projects.** Agile project planning offers some distinct advantages over the traditional, waterfall planning model. For example, waterfall models offer a rigid, set project plan, in which development occurs in a logical, sequential manner, following predetermined steps. For projects with a fixed set of goals and well-understood processes, waterfall planning works well. However, Agile is useful because for many projects, it recognizes the likelihood of scope and specification changes

occurring in the middle of the development cycle. Because it is an incremental, iterative planning methodology, Agile allows project teams to plan and execute elements of the project in shorter (1–4 week) segments, called Sprints. Finally, Agile recognizes that project success must be viewed through the eyes of the user, so it emphasizes the "voice of the customer" and the development of product features they value, rather than focusing solely on specifications. The flexibility of the methodology and its commitment to creating value for the customer make Agile a good alternative project planning approach.

2. **Recognize the critical steps in the Agile process as well as its drawbacks.** There are five steps in the Agile/Sprint process: (1) Sprint Planning—the work of the upcoming Sprint is identified and a Sprint Backlog is created; (2) Daily Scrums—meetings of the development team to synchronize their activities and plan for the next 24-hour window; (3) Development Work—the actual work in the Sprint is done during this stage and is often represented with a Burndown Chart; 4) Sprint Reviews—held at the end of the Sprint to inspect the work that was performed and make changes to the Product Backlog as needed; and 5) Sprint Retrospective—the meeting held to evaluate how the previous Sprint went, including corrective action for improving the process prior to the next Sprint.
3. **Understand the key features of the Extreme Programming (XP) planning process for software projects.** Extreme Programming (XP) is a software development technique that takes the customer responsiveness of Agile to extreme levels. Core principles of XP include an emphasis on keeping the programming code simple, reviewing it frequently, testing early and often, and working normal business hours. Two distinct elements in XP include the use of refactoring, which is the continuous process of streamlining the software design and improving code throughout development, rather than waiting for final product testing. The second feature of XP is the use of pair programming, in which sets of programmers work side-by-side to support each other's efforts. Pair programming promotes a collaborative process in creating software and helps maintain a constant emphasis on quality during development.
4. **Distinguish between critical path and critical chain project scheduling techniques.** As a result of systematic problems with project scheduling, Eli Goldratt developed the Critical Chain Project Management (CCPM) process. With CCPM, several alterations are made to the traditional PERT scheduling process. First, all individual activity slack, or "buffer," becomes

project buffer. Each team member responsible for her component of the activity network, creates a duration estimate free from any padding, that is, one that is based on a 50% probability of success. All activities on the critical chain and feeder chains (noncritical chains in the network) are then linked with minimal time padding. The project buffer is now aggregated and some proportion of that saved time (Goldratt uses a 50% rule of thumb) is added to the project. Even adding 50% of the saved time significantly reduces the overall project schedule while requiring team members to be less concerned with activity padding and more with task completion.

Second, CCPM applies the same approach for those tasks not on the critical chain. All feeder path activities are reduced by the same order of magnitude and a feeder buffer is constructed for the overall noncritical chain of activities.

Finally, CCPM distinguishes between its use of buffer and the traditional PERT use of project slack. With the PERT approach, project slack is a function of the overall completed activity network. In other words, slack is an outcome of the task dependencies, whereas CCPM's buffer is used as an a priori input to the schedule planning, based on a reasoned cut in each activity and the application of aggregated project buffer at the end.

5. **Understand how critical chain methodology resolves project resource conflicts.** Critical Chain Project Management assumes that the critical chain for a project requires first identifying resource conflicts and then sequencing tasks so as to eliminate these conflicts. Instead of employing early-start methods for networks, the CCPM approach emphasizes using late-start times, adding feeder buffers at the junction of feeder paths to the critical path, and applying an overall project buffer at the project level to be used as needed. All activities are sequenced so as to exploit resource conflicts, ensuring minimal delays between tasks and speeding up the overall project.
6. **Apply critical chain project management to project portfolios.** CCPM can also be applied at the project portfolio level, in which multiple projects are competing for limited project resources. Portfolio management first consists of identifying the maximum resource availability across all projects in a portfolio, prioritizing the projects for access to the constrained resource, and then sequencing other, noncritical project activities around the resources as they are available. The "drum resource" is the critical resource that constrains the whole portfolio. To buffer the projects that are sequenced to use the drum resources, CCPM advises creating capacity constraint buffers (CCBs) to better control the transition between projects as they queue to employ the

critical resource.