2

Planning for High-Quality Projects

2.1 The Hardest Time to Make a Plan Is When You Need It Most
When teams are under heavy pressure to commit, they must dig in their heels and insist on making a plan.

2.2 Make Two Kinds of Plans: Period and Product
The first is based on a period of time and concerns the way you plan to spend time during this period. The second is based on an activity, like developing a program or writing a report.

2.3 Make Product Plans for Every Major Task
Product plans help you to judge how much time the work will take and to track your progress.

2.4 Review Detailed Plans with Your Management
Few software organizations have a planning process that ensures that plans are complete, thoroughly reviewed, and properly approved, but these are what they need to manage their projects.

2.5 Everyone Loses with Incompetent Planning
With incompetent plans, the customers will receive late and more costly products, management will tie up excessive resources, and the developers will get a bad reputation.

2.6 Plans Must Meet Five Basic Requirements
A plan must be accessible, clear, specific, precise, and accurate.

2.7 When You Can't Plan Accurately, Plan Often
Don't let your plan be nibbled to death by small changes.

2.8 Plans Must Be Maintained
Even when projects are perfectly planned, changing conditions will often require plan adjustments.

2.1 THE HARDEST TIME TO MAKE A PLAN IS WHEN YOU NEED IT MOST

"This is an absolutely critical project and it must be completed in nine months." The division general manager was concluding his opening remarks at the kickoff for a project launch. As the Software Engineering Institute (SEI) coach, I had asked him to start the meeting by explaining why the business needed this product and how important it was to the company. As he told the development team, the company faced serious competition, and marketing could not hold the line for more than nine months. He then thanked me and the SEI for helping them launch this project and turned the meeting over to me.

After describing what the team would do during the next four days, I asked the marketing manager to speak. He described the new features needed for this product and how they compared with the current product and the leading competitor's new product. He explained that this new competitive machine had just hit the market and was attracting a lot of customer interest. He said that if this team did not develop a better product within nine months, they would lose all of their major customers. He didn't know how to hold the line.

On that happy note, we ended the meeting. The nine developers, the team leader, and I went to another room where we were to work. The team was scheduled to present its project plan to management in four days.

The developers were very upset. They had just finished one "death march" project and didn't look forward to another. After they calmed down a bit, Greg, the team leader, asked me, "Why make a plan? After all, we know the deadline is in nine months. Why waste the next four days planning when we could start to work right now?"

"Do you think you can finish in nine months?" I asked.

"Not a chance," he said.

"Well, how long will it take?"

"I don't know," he answered, "but probably about two years—like the last project."

So I asked him, "Who owns the nine-month schedule now?"

"The general manager," he said.

"OK," I said, "Now suppose you go back and tell him that nine months is way too little time. The last project took two years and this is an even bigger job. What will happen?"

"Well," Greg said, "he will insist that nine months is firm. Then I will tell him that we will do our very best."

"Right," I said, "that's what normally happens. Now who would own the nine-month schedule?"

"Oh," he answered, "I would."

"Suppose you promise a nine-month schedule and then take two or more years. Would that cause problems?"

"It sure would," Greg said. "Nobody would believe us again and our best customers would almost certainly jump ship."

"Listen," I explained, "management doesn't know how long this project will take, and neither do you. As long as you are both guessing, the manager will always win. The only way out of this bind is for you to make a plan and do your utmost to accomplish what management wants. Then, if you can't meet his deadline, you will understand why and be able to defend your schedule."

Then Ahmed, one of the development team members, jumped in. "OK, so we need a plan, but how can we make one if we don't even know the requirements?"

"Very good question," I answered, "but do you have to commit to a date?"

"Well, yes."

"So," I asked, "you know enough to commit to a date but not enough to make a plan?"

He agreed that this didn't make much sense. If they knew enough to make a commitment, they certainly ought to know enough to make a plan. "OK," he said, "but without the requirements, the plan won't be very accurate."

"So," I asked, "when could you make the most accurate plan?"

"Well, I suppose that would be after we have done most of the work, right?"

"That's right," I replied, "and that's when you least need a plan. Right now, your plan will be least accurate but now is when you most need a schedule that you can defend. To do this, you must have a detailed plan."

With that, the team agreed to make a plan.

This was one of our early Team Software Process (TSP) teams and the members had all been trained in the Personal Software Process (PSP). They then used PSP methods to make a plan that they believed was workable, and they then negotiated that plan with management. Throughout the project, they continued to use the PSP methods and, in the end, they delivered a high-quality product six weeks ahead of the 18-month schedule management had reluctantly accepted.

2.2 MAKE TWO KINDS OF PLANS: PERIOD AND PRODUCT

There are two kinds of planning. The first is based on a period of time, any calendar segment—a day, week, month, or year. A period plan concerns the way you plan to spend time during this period. The second kind of plan is based on an activity, like developing a program or writing a report. The products may be tangibles like programs or reports or intangibles like the knowledge you get from reading a textbook or the service you provide when working in an office.

To see the difference between period and product planning, consider the activity of reading a book. To plan this work, you would first estimate the time for the total job. For example, you might expect to take 20 hours to read the 20 chapters in an entire book. For the product plan, you would then schedule the time to do the reading, say one hour a week. The product plan for this task would then be the objective of reading the book chapters in 20 hours. The period plan would be the way you allocate reading time in one-hour weekly increments.

We all use both period and product plans in our daily lives. In business, for example, the two are related as shown in simplified form in Figure 2.1. The left half of the figure deals with the product-based tasks and the right half with the period-based tasks. These two are related as follows.

Corporate management provides funds for engineering and manufacturing to develop and produce products. Engineering develops products and releases them to manufacturing. Through the marketing group, manufacturing delivers these products to the customers, who pay for them. Engineering and manufacturing also provide product plans to finance and administration, who use the product plans to produce period plans for

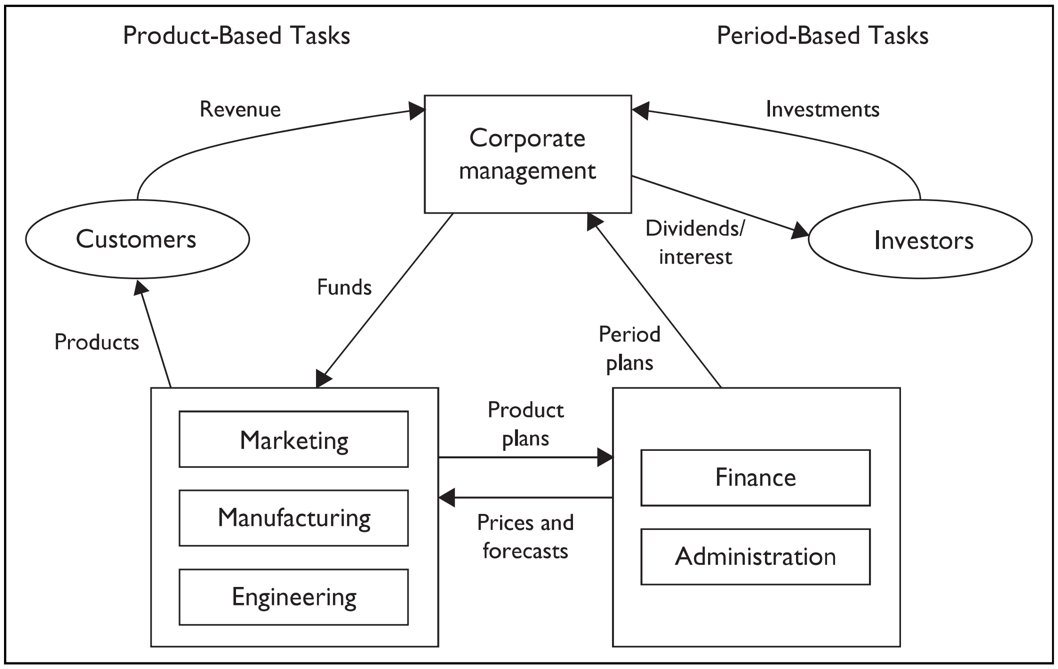

Figure 2.1 Relationship of product and period planning

quarterly and annual revenue and expense. Finance and administration give their plans to corporate management. They also give pricing and forecast information to engineering and manufacturing so they know how many products to build each month and what to charge the customers for them.

Corporate management decides what dividends and interest to pay the investors and how much new investment they need. When they know how much money they will have in the future, corporate management can then decide on the funds to provide engineering for future product development and manufacturing, completing the cycle.

In businesses, product plans are important because they tell marketing when they can expect new products to sell to customers. They also tell finance what development and manufacturing will cost. Finance can then determine the prices they need to charge and provide accurate period plans to corporate management. Management can then allot the money engineering and manufacturing will need. If the product plans are not accurate, the period plans will also be inaccurate. Management, marketing, the investors, and even the customers will be misinformed. This in turn could mean that the necessary funds would not be available to engineering and manufacturing when they are needed. Without adequate funds, engineering and manufacturing might have to cut back their plans. When they cut back their plans, *you* could be laid off.

While this planning cycle may sound theoretical and may take several years, the threat is real. The key to job security in any business is the financial health of the organization. And a key to business financial health is accurate period and product plans. It is therefore important that engineers know how to make sound period and product plans.

While period and product plans must be related, they are different. Your work schedule, your income, and your other daily activities are governed by time periods. That is, you sleep, eat, work, and rest during certain periods of the day or week. Your expenses and income are governed by weekly, monthly, and annual schedules. You thus live in a world with period activities, rules, and constraints.

The principal purpose of your work, however, is to produce products and services of value to others. The cost, schedule, and quality of these goods or services are thus important. Since your work will be on products and your life will be in periods, both period and product plans are important to you. You cannot make a competent plan for either one without also planning the other.

2.3 MAKE PRODUCT PLANS FOR EVERY MAJOR TASK

Some years ago while at IBM, I was put in charge of a large software development department with many projects. Most of the projects were seriously late and senior management was very upset. My job was to straighten out the mess. The first thing I did was to review the major projects. To my surprise, none of them had any documented plans. I immediately insisted the engineers produce plans for all their projects. Never having prepared formal plans before, it took them a couple of months to do it. We even set up a special class on project planning. After they had made plans, however, they could establish realistic schedules for their work.

The act of making plans had a dramatic effect. This development group had never before delivered a product on time. Starting with their new plans, however, they did not miss a single date for the next two and a half years. And they have been planning their work ever since. Planning is a critical part of a software engineer's job, and to be an effective engineer, you need to know how to make plans. The key is practice, so to get the most practice, start making plans now and continue to do so for all your future projects.

I suggest that you develop product plans for all your projects or major tasks: writing a program, reading a textbook, or preparing a report. The product plan will help you judge how much time the work will take and when you will finish. Plans also help you track progress while doing the work.

When engineers work on development teams, they need to plan their personal work. Planning provides a sound basis for committing to completion dates, and it allows engineers to coordinate their work on joint products. Their individual product plans enable them to commit dates to each other for their interdependent tasks and to consistently meet those commitments.

Businesses use product plans for the same reasons you will: to plan and manage their work. A well-made plan includes a project cost estimate. Estimates are essential for development contracts because customers often need to know the price in advance. Estimates are also necessary when developing products. Project cost is a major part of product price and must be kept low enough for the price to be competitive in the marketplace.

Engineers also use product plans to understand project status. With reasonably detailed and accurate plans, they can judge where a project stands against the plan. They can see if they are late and need help or if they will have to delay the schedule. They may even be ahead of schedule and be able to help their teammates or deliver early. When engineers make plans, they can better organize their time and avoid last-minute crises. They are then less likely to make mistakes and will generally produce better products. Because plans are so important, you need to know how to make accurate plans. You also need to know how to compare these plans with actual results so you can learn to make better plans.

The first step in producing a product plan is to get a clear definition of the product you plan to produce. While this seems obvious, it is surprising how often people dive into the mechanics of developing a product before they define what they are trying to do. Only after you know what you want to do, should you start thinking about how to do it. That is when to start planning.

A properly produced product plan includes three things:

- The size and important features of the product to be produced,
- An estimate of the time required to do the work, and
- A projection of the schedule.

The product could be a working program, a program design, or a test plan. A product plan thus identifies the product to be produced and contains estimates of the product size, the hours to do the work, and the schedule. More complex products require more sophisticated planning and many kinds of information, such as responsibility assignments, staffing plans, product or process specifications, dependencies on other groups, or special testing or quality provisions.

2.4 REVIEW DETAILED PLANS WITH YOUR MANAGEMENT

Software development plans are often incomplete and inaccurate. During the 27 years when I worked at IBM, we once needed a critical new function for the OS/360 programming system. The engineering estimate was $175,000. Naively, that is all the funding I requested. Some months later, the developers found that the work would cost $525,000. They had omitted many necessary tasks from their original plan. They had forgotten documentation; testing; the integration, build, and release processes;

and quality assurance. Sure enough, however, the coding and unit test costs were about $175,000. They had made a pretty good estimate, but their plan was painfully (for me) incomplete. I had to make up the difference out of department funds.

The problem is that few software organizations have a planning process that ensures that plans are complete, thoroughly reviewed, and properly approved. Even worse, few software developers have the knowledge and experience to make sound plans. When you start a project, management typically states very clearly when they want the job done, but they are not usually very clear about much else. It is not that they are bad, lazy, or incompetent; it is just that, except for the schedule, most requirements are complex and cannot be easily described. By emphasizing the schedule, management gives the impression that it is their highest-priority concern. But, although the schedule is important, you must also address all of management's stated and implied goals. In addition, while doing this, you must do your best to meet management's desired schedule.

What management really wants is a completed project *now*, at no cost. Anything else is a compromise. However, because they know that development takes time, they will push for the most aggressive schedule that you and your team will accept as a goal. Not unreasonably, they believe that projects with the shortest schedules finish before ones with longer schedules. Therefore, they will keep pushing until they believe that the schedule is the shortest one you will agree to meet. As developers, however, we are responsible for doing the work. With an impossibly short schedule, it is difficult if not impossible to make a usable plan. Then, without a plan, we are generally in such a rush to code and test that we cut corners and don't do as good a job as we could. Such projects generally take much more time than they would with a realistic plan.

Typically, when management asks for an aggressive date, the developers tell them that this date doesn't allow enough time to do the work. Management then insists that the date is firm, and the team generally caves in and agrees to do its best. Such teams start out in trouble and almost always end up in trouble. As Greg's team found in Section 2.1, the best answer to this problem is to make a detailed plan and to review it with management. Because most managers want a schedule you *can* meet, they will negotiate a thoughtfully made plan with you. If you present a convincing case, they will then end up agreeing to your schedule. To have such a debate, however, you must know how to make a plan and how to defend it to management. PSP training provides these skills.

PSP training will help you to build the needed planning skills. Planning is the first step in the PSP for three reasons. First, without good plans you cannot effectively manage even modestly sized software projects. Second, planning is a skill that you can learn and improve with practice. Third, good planning skills will help you to do better software work. Later, on a team, the TSP shows you how to prepare for and handle the plan negotiations with management.

2.5 EVERYONE LOSES WITH INCOMPETENT PLANNING

In software engineering, as in other fields, our role as developers is to devise economical and timely solutions to our employer's needs. To do this, we must consider costs and schedules. The connection between cost estimating, scheduling, and the planning process can best be illustrated by an example. Suppose you want to put an addition on your home. After deciding what you want and getting several bids, most of which are around $24,000, you pick a builder who offers to do the job in three months for $20,000. Although this is a lot of money, you need the extra space and can arrange for a home-equity loan. You then sign an agreement and the builder starts the work. After about a month into the job, the builder tells you that, because of unforeseen problems, the job will take an extra month and cost an additional $4,000.

This presents you with several problems. First, you badly need the space, and another month of delay is a great inconvenience. Second, you have already arranged for the loan and don't know where you can get the extra $4,000. Third, if you get a lawyer and decide to fight the builder in court, all the work will stop for many months while the case is settled. Fourth, it would take a great deal of time and probably cost even more to switch to a new builder in the middle of the job.

After considerable thought, you decide that the real problem is that the builder did a sloppy job of planning. Although you do not know precisely what went wrong, the builder probably got low bids on some of the major subcontracts, such as the plumbing, plastering, or painting. You can endlessly debate what the job should cost, but essentially the problem was caused by poor planning. If you had originally been given the $24,000 price, you could have decided then whether to proceed with that builder and how to finance the work. The odds are good that at this point you will try to negotiate a lower price but continue with the current builder. Because the other bids were close to $24,000, you know this is a pretty fair price. You would not use this builder again, however, and would probably not recommend him to anyone else.

The problem with incompetent planning is that everybody loses: customers receive late and more costly products, management must tie up more resources, and the developer gets a bad reputation. To be successful, businesses must meet their commitments. To do our part, we must produce plans that accurately represent what we will do.

Planning is serious business. It defines commitments and supports business decisions. Well-thought-out plans will help you to make commitments that you can meet, and enable you to accurately track and report your progress. Personal planning skill will be even more important when you work on a development team. The overall team plan is most likely to be realistic when it is built from competently made team-member plans.

"The project plan defines the work and how it will be done. It provides a definition of each major task, an estimate of the time and resources required, and a framework for management review and control. The project plan is also a powerful learning vehicle. When properly documented, it is a benchmark to compare with actual performance. This comparison permits the planners to see their estimating errors and to improve their estimating accuracy."[1] Plans typically are used as the following:

- A basis for agreeing on the cost and schedule for a job
- An organizing structure for doing the work
- A framework for obtaining the required resources
- The standard against which to measure job status
- A record of what was initially committed

The connection between plans and commitments is extremely important. Every project starts as a new endeavor. At the outset, the project must be created out of thin air. New projects typically start with no staff. A manager, user, or customer must commit funds, and some workers and suppliers must be convinced to participate in the work. For substantial projects, management's

1. Watts S. Humphrey. 1989. *Managing the Software Process.* Reading, MA: Addison-Wesley.

first step is to assemble a planning and proposal team and produce an overall plan. Without a clear and convincing plan, they will not be able to get funding, hire the staff, and arrange for all the facilities, supplies, and other support needed to do the work. Nobody wants to pay for an undefined job, and few people will work on a project that has unclear objectives. Because an accurate plan is the essential first step in creating a successful project, planning is an important part of every project.

2.6 PLANS MUST MEET FIVE BASIC REQUIREMENTS

In producing a plan, the result must meet certain requirements. The five basic requirements for a plan are that it be accessible, clear, specific, precise, and accurate.

Is It Accessible?

Accessibility is best illustrated by the case of a company that was fighting a major U.S. government lawsuit. At one point, they were required to provide all of the historical documents that related to a particular topic. Because the company was large, this material was scattered among thousands of files. The cost of searching all those files would have been prohibitive, so the company gave the plaintiff all the files that could possibly contain any of the desired data. Even though these files undoubtedly contained material the company didn't want released, they judged this problem to be minor compared to the costs of a detailed document review. Although the plaintiff knew that what he wanted was somewhere in the many thousands of pages of files, he might as well not have had it. In practical terms, it was inaccessible.

To be accessible, a plan must provide the needed information so that you can find it; it must be in the proper format; and it must not be cluttered with extraneous material. Although having complete plans is important, voluminous plans are unwieldy. You need to know what is in the plan and where it is. You should be able to quickly find the original schedule and all subsequent revisions. The defect data should be clear, and the program size data must be available for every program version. To be most convenient, these data should be in a prescribed order and in a known, consistent, and nonredundant format.

Is It Clear?

I find it surprising that even experienced developers occasionally turn in PSP homework that is sloppy and incomplete. Some items are left blank, others are inconsistent, and a few are obviously incorrect. Sometimes, even incomprehensible notes are jotted in the margins. Such work is not data and should be rejected. If the data are not complete and unmistakably clear, they cannot be used with confidence. If they cannot be used with confidence, there is no point in gathering them at all.

Is It Specific?

A specific plan identifies what will be done, when, by whom, and at what costs. If these items are not clear, the plan is not specific.

Is It Precise?

Precision is a matter of relating the unit of measure to the total magnitude of the measurement. If, for example, you analyzed a project that took 14 programmer years, management would not be interested in units of minutes, hours, or probably even days. In fact, programmer weeks would probably be the finest level of detail they could usefully consider. For a PSP job that takes five hours, however, units of days or weeks would be useless. Conversely, units of seconds would be far too detailed. Here you are probably interested in time measured in minutes.

To determine an appropriate level of precision, consider the error introduced by a difference of one in the smallest unit of measure. A project that is planned to take 14 programmer years would require 168 programmer months. An uncertainty of one month would contribute an error of at most 0.6 percent. In light of normal planning errors, this is small enough to be quite acceptable. For a five-hour project, an uncertainty of one minute would contribute a maximum error of about 0.33 percent. A unit of measure of one minute would thus introduce tolerable errors, whereas units of an hour or even a tenth of an hour would probably be too gross.

Is It Accurate?

Although the other four points are all important, accuracy is crucial. A principal concern of the planning process is producing plans with predictable accuracy. As you plan the PSP work, do not be too concerned about the errors in each small task plan as long as they appear to be random. That is, you want to have about as many overestimates as underestimates. These are called **unbiased estimates.** As you work on larger projects or participate on development teams, the small-scale errors will balance each other out and the combined total will be more accurate.

2.7 WHEN YOU CAN'T PLAN ACCURATELY, PLAN OFTEN

A common belief is that projects get into trouble because their requirements changed. This is an excuse. Requirements always change. Usually, the problem is with the way these changes are managed. When faced with lots of little requirements changes, it takes considerable effort to keep the plan up to date. However, if you don't, you will be working without a plan and can no longer track your work or estimate when the job will be done. While this often happens to projects, the problem is with how the project was managed—not with the requirements changes. Often, the requirements will change so often and in such small steps that you will have to constantly reassess your plan.

With a dynamic planning process, teams can assess the impact of each change and only agree to changes when management understands and agrees to the necessary schedule and resource adjustments. The team can then know the cost of each requirements change and can negotiate any needed schedule or resources requirements before committing to the change. While this sounds easy in theory, it is often exceedingly difficult in practice.

Dynamic planning will protect you from requirements creep. If you do not examine the impact of every change, you will be inundated with small changes. In effect, by accepting changes without adjusting the schedule or resources, you are telling the customer and management that changes are free. That both misleads them and invites more changes. If you do not control every change, your project will be overwhelmed with changes.

Act as if there is no such thing as a free change. This is, of course, an overreaction. There is a fine line between requirements changes and requirements clarifications. As the customer's understanding of the product matures, the only constraint on adding new functions and features is your ability to deliver. If you do not make that constraint known, you will mislead the business. The best way to do this is with dynamic planning. It is the only way to protect yourself from being nibbled to death by small changes.

The longer you work to a detailed plan, the more you will need to change it. This does not mean that you will actually change the plan, but that you will refine it. As the work progresses, the team members will better understand the work and be better able to make more detailed plans. While these small plan changes will usually be consistent with the original plan, this dynamic working plan will gradually diverge from the plan you made during the launch.

Suppose that, after a few weeks of work, one team member completes her planned tasks and is ready to start on some new ones. If the rest of the team is still following its prior plan, it might not make sense to conduct a team relaunch. Rather than have this one developer work without a plan, she could add some near-term tasks to her current plan and continue working. To allow for this possibility, it is generally a good idea to extend the team's planning well beyond the planned relaunch date. Rather than plan all of the future work in detail, however, the later tasks could be kept in larger chunks and only refined when needed. In fact, developers often find it desirable to defer detailed planning for their larger tasks until shortly before they start to work on them.

While it is often possible to keep extending the current plan, it is not a good idea to defer the team relaunch for too long. The relaunch is needed not just to update the plan, but to update the team's goals and to review the team's risks and role assignments. It is also a good time to assess the work that has been done and to agree on how to do better work during the next project cycle.

2.8 PLANS MUST BE MAINTAINED

Although planning is essential for well-run projects, detailed plans are generally accurate for only brief periods of time. As developers work, they learn more about the job and its many parts, and they see how long it takes to do various kinds of tasks. They also learn more about the product and have a better idea of how to develop it. Another factor affecting plans is that the developers' organizations and projects will change and evolve. Thus, even when projects are perfectly planned, changing conditions will often require plan adjustments. So, unless the members regularly adjust their plans, these plans will become less and less relevant to their work.

Even without external changes, the team's detailed plans will become inaccurate. Some developers will finish tasks ahead of their plans, and others will fall behind. There will also be estimating errors and overlooked tasks. The more detailed the plans and the longer they are used, the more inaccurate they will become. This is a problem because the team members' work must be synchronized. They must come together for design reviews, inspections, and management reviews. They must also coordinate to produce the team's products and to meet project milestones. As the workload becomes unbalanced, the schedule for these synchronization points will then be determined by the developers who are furthest behind. Periodic replanning is the easiest and quickest way to address this imbalance problem.

Managers get nervous when developers update their plans. They intuitively expect that every replan will extend the schedule. However, once teams learn to make accurate plans and to work to these plans, they can replan without significantly changing their committed schedules. If the original plan was competently made, if it was based on historical size and task-hour data, and if there were no significant external changes, the baseline plan should be pretty accurate. Then, the team's plan changes will only involve the normal estimating fluctuations at the detailed level plus any resource and requirements variations.

At the detailed level, team plans must change on a regular basis. If the plans are based on average historical data, about half of the developers should finish their tasks early and the other half late. However, on average the team's overall plan will still be consistent with the baseline plan. The problem is that some developers will occasionally have several consecutive tasks that take longer than planned. They will then fall behind schedule and, unless their team reassigns some of their tasks, these normal workload fluctuations could lead to serious workload imbalances. Teams should regularly identify the members who are early and can handle additional workload and those who are behind and need tasks offloaded. If teams do not keep their detailed plans in balance, some members will eventually fall so far behind that they will delay the project.

SOURCES

2.1: From *PSP^{SM}: A Self-Improvement Process for Software Engineers,* Chapter 1

2.2: From *Introduction to the Personal Software Process^{SM}*, Chapter 4

2.3: From *Introduction to the Personal Software Process^{SM}*, Chapter 5

2.4: From *PSP^{SM}: A Self-Improvement Process for Software Engineers,* Chapter 4

2.5: From *PSP^{SM}: A Self-Improvement Process for Software Engineers,* Chapter 4

2.6: From *PSP^{SM}: A Self-Improvement Process for Software Engineers,* Chapter 7

2.7: From *TSP^{SM}: Leading a Development Team,* Chapter 8

2.8: From *TSP^{SM}: Coaching Development Teams,* Chapter 18