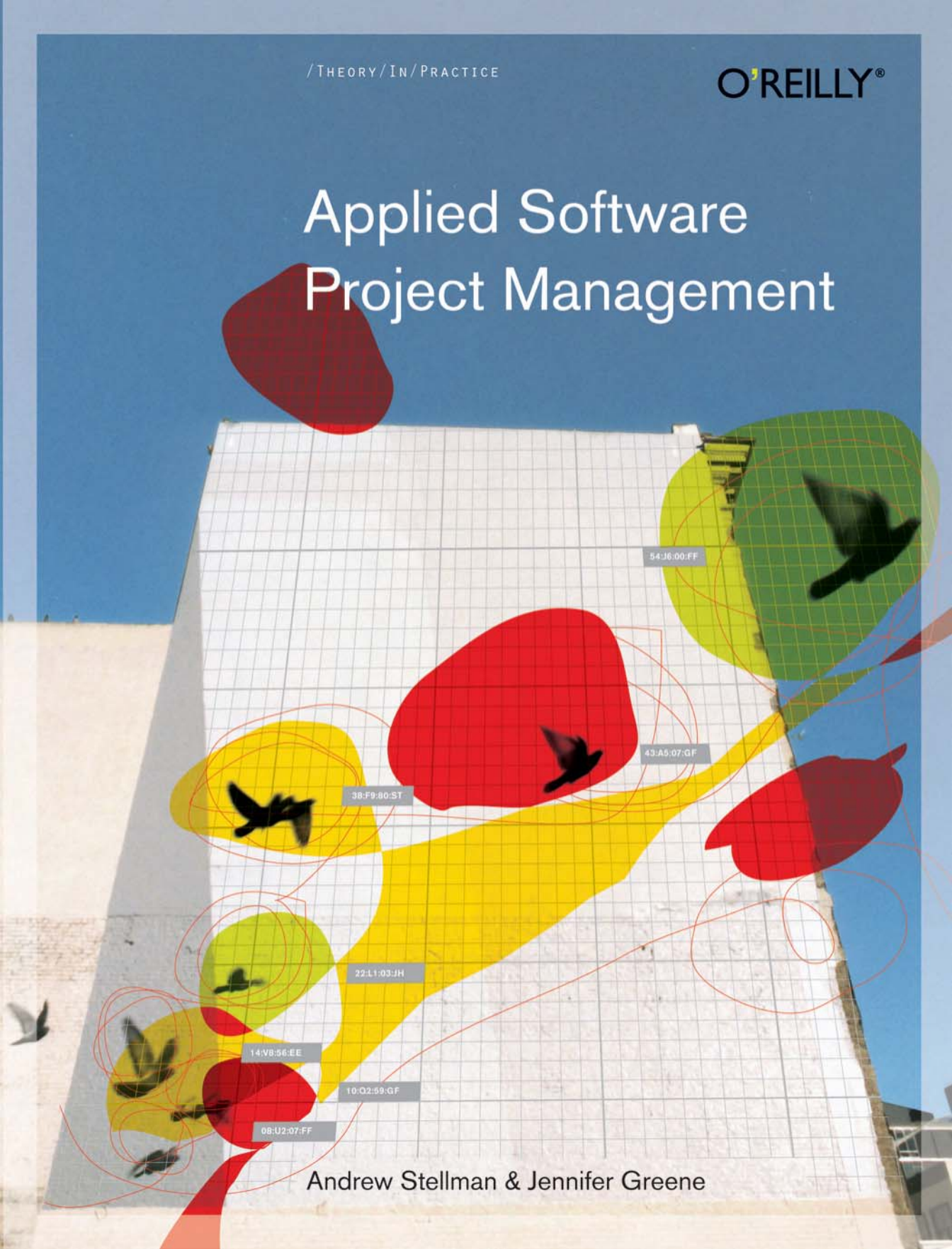
/THEORY/IN/PRACTICE

O'REILLY®

Applied Software Project Management



Andrew Stellman & Jennifer Greene

CHAPTER THREE

Estimation

MANY PEOPLE HAVE REFERRED TO ESTIMATION AS A "BLACK ART." This makes some intuitive sense: at first glance, it might seem that estimation is a highly subjective process. One person might take a day to do a task that might only require a few hours of another's time. As a result, when several people are asked to estimate how long it might take to perform a task, they will often give widely differing answers. But when the work is actually performed, it takes a real amount of time; any estimate that did not come close to that actual time is inaccurate.

To someone who has never estimated a project in a structured way, estimation seems little more than attempting to predict the future. This view is reinforced when off-the-cuff estimates are inaccurate and projects come in late. But a good formal estimation process, one that allows the project team to reach a consensus on the estimates, can improve the accuracy of those estimates, making it much more likely that projects will come in on time. A project manager can help the team to create successful estimates for any software project by using sound techniques and understanding what makes estimates more accurate.

Elements of a Successful Estimate

A sound estimate starts with a *work breakdown structure* (WBS). A WBS is a list of tasks that, if completed, will produce the final product. The way the work is broken down dictates how it will be done. There are many ways to decompose a project into tasks. The project can be broken down by feature, by project phase (requirements tasks, design tasks, programming tasks, QA tasks, etc.), or by some combination of the two. Ideally, the WBS should reflect the way previous projects have been developed.

A useful rule of thumb is that any project can be broken down into between 10 and 20 tasks. For large projects (for example, a space shuttle), those tasks will be very large ("Test the guidance system"); for small projects (like writing a simple calculator program), the tasks are small ("Build the arithmetic object that adds, multiplies, or divides two numbers"). The team must take care in generating the WBS—if the tasks are incorrect, they can waste time going down a wrong path.

Once the WBS is created, the team must create an estimate of the effort required to perform each task. The most accurate estimates are those that rely on prior experience. Team members should review previous project results and find how long similar tasks in previous projects took to complete. Sources of delays in the past should be taken into account when making current estimates. Postmortem reports (see Chapter 8) are a good source of this information.

No estimate is guaranteed to be accurate. People get sick or leave the organization; teams run into unforeseen technical problems; the needs of the organization change. The unexpected will almost certainly happen. Therefore, the goal of estimation is not to predict the future. Instead, it is to gauge an honest, well-informed opinion of the effort required to do a task from those people in the organization who have the most applicable training and knowledge.

If two people widely disagree on how long a task will take, it's likely that the source of that disagreement is that each person made different assumptions about details of the work product or the strategy for producing it. In other words, any disagreement is generally about what is required to perform the task itself, not about the effort required to complete it. For example, given the same vision and scope document for a tool that sets the computer clock, two different developers might come up with wildly different estimates. But it might turn out that one developer assumed that the implementation would have a simple command-line interface, while the other assumed that there would be a complete user interface that had to integrate tightly with the operating system's control panel. By helping the programmers discuss these assumptions and come to a temporary resolution about their differences, the project manager can help them agree on a single estimate for the task.

A project manager can help the team create more accurate estimates by reducing the uncertainty about the project. The most effective way to do this is to do a thorough job creating a vision and scope document (see Chapter 2)—the more accurate and detailed it is, the more information the team has to work with when generating their estimate. The project manager can also ensure that the team has reached a consensus on the tasks that must be performed. Finally, the project manager can lead the team in a discussion of assumptions.

Assumptions Make Estimates More Accurate

Once the team has agreed upon a WBS, they can begin to discuss each task so they can come up with an estimate. At the outset of the project, the team members do not have all of the information they need in order to produce estimates; nevertheless, they need to come up with numbers. To deal with incomplete information, they must make assumptions about the work to be done. By making assumptions, team members can leave placeholders for information that can be corrected later, in order to make the estimate more accurate.

For the estimates to be most effective, the assumptions must be written down. Important information is discovered during the discussion that the team will need to refer back to during the development process, and if that information is not written down, the team will have to have the discussion all over again. If an assumption turns out to be incorrect, the schedule will need to be adjusted; they will be able to point to the exact cause of the delay by showing that a documented assumption turned out to be incorrect. This will help the project manager explain any resulting schedule delay to others in the organization and avoid that source of delays in the future. The assumptions also provide a way to keep a record of team decisions, share those decisions with others, and find errors in their decisions. (The assumptions should be added to the "Assumptions" section of the vision and scope document—see Chapter 2.)

The team should hold a brainstorming session to try to identify as many assumptions as possible. The bigger the list of assumptions, the lower the overall risk for the project. A project manager may get better results from this session by helping the team see how these assumptions can work to their benefit. Any software engineer who has had a bad experience with an estimate that has turned out to be inaccurate will appreciate the value

of assumptions: they serve as a warning to the rest of the organization that the estimate is contingent on the assumptions being true. If even one of these assumptions turns out to be incorrect, the team cannot be "blamed" for the incorrect estimate that resulted.

While identifying assumptions is a skill that improves with experience, there are a set of questions that can help a novice team member figure out what assumptions he or she needs to make in order to properly estimate the software. The project manager (or a moderator in a Wideband Delphi session—see below) can use these questions to help lead the discussion to identify the assumptions:

- Are there project goals that are known to the team but not written in any documentation?
- Are there any concepts, terms, or definitions that need to be clarified?
- Are there standards that must be met but will be expensive to comply with?
- How will the development of this project differ from that of previous projects? Will there be new tasks added that were not performed previously?
- Are there technology and architecture decisions that have already been made?
- What changes are likely to occur elsewhere in the organization that could cause this estimate to be inaccurate?
- Are there any issues that the team is known to disagree on that will affect the project?

The last bullet point is especially important. If one team member believes that the project will go down one path while another team member believes the project will go down a different path, the estimates could vary significantly, depending on which team member is correct. For example, one team member may think that a certain off-the-shelf component should be used because that is cheaper than building it, while another team member may believe that they must build that component themselves because they cannot locate one for sale which suits their particular needs. Instead of reaching an impasse, the team can make an assumption—either assume that they will buy the component, or assume that they will build it—which will enable them to move forward with the estimate. It should be easier to reach an agreement at this point because it is not the final decision. By writing down the assumption, the team keeps a record of the disagreement and leaves open the possibility that this will change in the future. The written assumption will be especially useful later while doing a risk assessment for the project plan because there is a risk that the assumption is incorrect.

In other words, assumptions can help find a compromise to resolve disagreements. If two team members disagree, the team can agree to write down an assumption to temporarily resolve the issue for the purposes of the estimate. It's much easier to get people to agree on one answer temporarily by agreeing to revisit the issue later.

Discussing and writing down the assumptions in a team setting helps the team to identify potential roadblocks. For example, the team may have a genuine disagreement about whether or not to develop a user interface for their clock-setting application. The assumption allows the team to reach a temporary decision, knowing that the final decision is still

open. Writing down the assumption allows the team to go back and revise the estimate later if it turns out the assumption is wrong—which means that it is vital that everyone understands that the assumptions are allowed to be incorrect. That way, if the team estimated that they would build a command-line program but later the decision was made to go with a full user interface, everyone will be able to explain why the schedule is delayed.

Another benefit of discussing assumptions is that it brings the team together very early on in the project to make progress on important decisions that will affect development. It's all too common for a developer to make estimates after reading the vision and scope document but before ever talking to anyone about the details of the project. Even if she writes down her assumptions, she has almost certainly missed many others. A moderated discussion of assumptions gives the project team a very effective forum to discuss the unknowns of the project. Identifying unknowns eliminates the source of many inaccuracies in the estimates.

One side effect of writing down the assumptions is that it puts pressure on the senior managers to allow the project to be estimated again if any of the assumptions prove to be incorrect. This is why the project manager should plan on having the vision and scope document updated to include any new assumptions that were identified during the estimation session. This gives the stakeholders and management a chance to review those assumptions and accept or reject them very early on, before they have had a chance to interfere with the development of the software. By having the senior managers review the assumptions, a project manager can reduce a source of future project delays.

Distrust Can Undermine Estimates

Estimates can either be a source of trust or distrust between the project team and their managers. If the team knows that they are given the opportunity to fully justify their estimates, and that they will be allowed to reestimate if the scope of the project changes, that they won't be punished for making an honest mistake in estimation, then each team member will work very hard to produce accurate estimates. In this case, estimation can be an effective tool for team motivation. Estimates are most accurate when everyone on the project team feels that he was actively part of the estimation process. Every team member feels a personal stake in the estimates, and will work very hard to meet any schedule based on those estimates.

Estimation is, by its nature, a politically charged activity in most organizations. When a team is asked to create estimates for work, they are essentially being asked to define their own schedule. Stakeholders need the project completed but usually do not have software engineering experience, so they may not be equipped to understand why a project will take, say, six months instead of three. For this reason, project managers must take care to make the estimation process as open and honest as possible so that the stakeholders can understand what's going on.

It is common for nontechnical people to assume that programmers pad their estimates. They often have a "rule" by which they cut off a third or half of any estimate that they hear, and expect that to be the "real" deadline. They often feel, fairly or not, that the engineering team

is "putting one over" on them, mostly because the entire engineering process is, to them, a mystery. This lack of trust causes engineers to automatically pad their estimates, because they know they won't be given enough time otherwise. And even when the situation is not this bad (although it often is), some environment of distrust still exists to a lesser extent in many organizations.

In many of these organizations, there are some kinds of estimates—especially those for quality and requirements tasks—that are particularly likely to not be taken seriously. Senior managers are often willing to take the programmers' estimates at face value, even when those estimates are clearly padded. This is because, to them, programming is opaque: managers and stakeholders don't understand how code is written, so they assume that all programming tasks are difficult. They are more likely to trust programmers' estimates, even when those estimates are highly padded. Requirements analysts, on the other hand, often produce specifications using nothing more than a word processor (and many elicitation sessions—see Chapter 6). A manager or stakeholder is much more likely to trivialize that work and distrust the estimate because he (incorrectly) feels that he has an intuitive grasp on the work being done. Even worse, in some organizations there is a "rule" that testing should always take exactly one-third (or some other fixed ratio) of the programming time, which causes the testing effort to be shortchanged by only allowing exactly that much time for it instead of the actual amount of time testing would require.

Distrust in a software organization can be a serious, endemic problem. It starts with a kernel of distrust between management and the engineering team; the distrust grows until management simply won't accept the team's estimates. For example, a senior manager may decide that the team plans to spend too much time testing the software, even though the team reached consensus and all team members stand behind the estimates. A project manager must be especially careful to explain this and support that consensus when senior managers start to pick apart the team's estimates. If deadlines are handed down that do not allow enough time for the team to complete the work, it can lead to serious morale problems—and the project manager will be blamed for the delay, often by the same people who caused it in the first place.

An important part of running successful software projects is reaching a common understanding between the engineers, managers, and stakeholders. One way to do that is with a consistent set of practices. This allows the engineers' work to be transparent to the rest of the organization. Similarly, the managers' and stakeholders' needs and expectations must be transparent to the engineers. By having key managers attend the estimation session, a project manager can show them that the estimates are made systematically, using an orderly and sensible process, and that they are not just made up on a whim. When the team is having trouble reaching convergence on a task, team members should bring up examples of past results for tasks of similar size and complexity. This transparency helps everyone present (especially the observers) understand why these estimates come out as they do.

Wideband Delphi Estimation

The *Wideband Delphi* estimation method was developed in the 1940s at the Rand Corporation as a forecasting tool. It has since been adapted across many industries to estimate many kinds of tasks, ranging from statistical data collection results to sales and marketing forecasts. It has proven to be a very effective estimation tool, and it lends itself well to software projects.*

The Wideband Delphi estimation process is especially useful to a project manager because it produces several important elements of the project plan. The most important product is the set of estimates upon which the project schedule is built. In addition, the project team creates a work breakdown structure (WBS), which is a critical element of the plan. The team also generates a list of assumptions, which can be added to the vision and scope document.

The discussion among the team during both the kickoff meeting and the estimation session is another important product of the Delphi process. This discussion typically uncovers many important (but previously unrecognized) project priorities, assumptions, and tasks. The team is much more familiar with the work they are about to undertake after they complete the Wideband Delphi process.

Wideband Delphi works because it requires the entire team to correct one another in a way that helps avoid errors and poor estimation. While software estimation is certainly a skill that improves with experience, the most common problem with estimates is simply that the person making the estimate does not fully understand what it is that he is estimating. He may be an experienced software engineer, but if he has not fully explored all of the assumptions behind the estimate, the estimate will be incorrect. Delphi addresses this problem through the discussion of assumptions and the generation of consensus among the estimation team members.

> NOTE
>
> **The Wideband Delphi process described here depends on a vision and scope document. It is possible to estimate a project without a vision and scope document, instead relying solely on the project team's understanding of the organization's needs. However, if there is no vision and scope document for the project, most project managers will find that writing one will improve the project far more than trying to estimate without it. (Chapter 2 describes how to develop a vision and scope document as part of the software project planning activities.)**

* The repeatable process for Wideband Delphi was developed in the 1990s by Mary Sakry and Neil Potter of The Process Group. Potter and Sakry offer a software estimation workshop based on their process. Information on their training products can be found at *http://www.processgroup.com*.

The Delphi Process

To use Wideband Delphi, the project manager selects a moderator and an estimation team with three to seven members. The Delphi process consists of two meetings run by the moderator. The first meeting is the kickoff meeting, during which the estimation team creates a WBS and discusses assumptions. After the meeting, each team member creates an effort estimate for each task. The second meeting is the estimation session, in which the team revises the estimates as a group and achieves consensus. After the estimation session, the project manager summarizes the results and reviews them with the team, at which point they are ready to be used as the basis for planning the software project. The script in Table 3-1 describes the Wideband Delphi process.

TABLE 3-1. Wideband Delphi script

Name	**Wideband Delphi script**
Purpose	A project team generates estimates and a work breakdown structure.
Summary	A repeatable process for estimation. Using it, a project team can generate a consensus on estimates for the completion of the project.
Work Products	*Input* Vision and scope document, or other documentation that defines the scope of the work product being estimated *Output* Work breakdown structure (WBS) Effort estimates for each of the tasks in the WBS
Entry Criteria	The following criteria should be met in order for the Delphi process to be effective: • The vision and scope document (or other documentation that defines the scope of the work product being estimated) has been agreed to by the stakeholders, users, managers, and engineering team. If no vision and scope document is available, there must be enough supporting documentation for the team to understand the work product. • The kickoff meeting and estimation session have been scheduled (each at least two hours). • The project manager and the moderator agree on the goal of the estimation session by identifying the scope of the work to be estimated.
Basic Course of Events	1. *Choosing the team.* The project manager selects the estimation team and a moderator. The team should consist of three to seven project team members. The team should include representatives from every engineering group that will be involved in the development of the work product being estimated. 2. *Kickoff meeting.* The moderator prepares the team and leads a discussion to brainstorm assumptions, generate a WBS, and decide on the units of estimation. 3. *Individual preparation.* After the kickoff meeting, each team member individually generates the initial estimates for each task in the WBS, documenting any changes to the WBS and missing assumptions. 4. *Estimation session.* The moderator leads the team through a series of iterative steps to gain consensus on the estimates. At the start of the iteration, the moderator charts the estimates on the whiteboard so the estimators can see the range of estimates. The team resolves issues and revises estimates without revealing specific numbers. The cycle repeats until either no estimator wants to change his or her estimate, the estimators agree that the range is acceptable, or two hours have elapsed. 5. *Assembling tasks.* The project manager works with the team to collect the estimates from the team members at the end of the meeting and compiles the final task list, estimates, and assumptions. 6. *Reviewing results.* The project manager reviews the final task list with the estimation team.

TABLE 3-1. Wideband Delphi script (continued)

Name	Wideband Delphi script
Alternative Paths	1. During Step 1, if the team determines that there is not enough information known about the project to perform an estimate, the script ends. Before the script can be started again, the project manager must document the missing information by creating or modifying the vision and scope document (see Chapter 2). 2. During either Step 1 or 3, if the team determines that there are outstanding issues that must be resolved before the estimate can be made, they agree upon a plan to resolve the issues and the script ends.
Exit Criteria	The script ends after the team has either generated a set of estimates or has agreed upon a plan to resolve the outstanding issues.

Choosing the team

Picking a qualified team is an important part of generating accurate estimates. Each team member must be willing to make an effort to estimate each task honestly, and should be comfortable working with the rest of the team. Estimation sessions can get heated; a team that already has friction will find that it runs into many disagreements that are difficult to resolve. The free flow of information is essential, and the project manager should choose a group of people who work well together. The estimators should all be knowledgeable enough about the organization's needs and past engineering projects (preferably similar to the one being estimated) to make educated estimates.

The moderator should be familiar with the Delphi process, but should not have a stake in the outcome of the session, if possible. Project managers are sometimes tempted to fill the moderator role, but this should be avoided (if at all possible) because the project manager should ideally be part of the estimation team. This is because the PM needs to take an active role in the discussion of the assumptions. She usually has a perspective on the project priorities that some of the engineers, stakeholders, and users do not see at first.

The role of the moderator is to listen to the discussion, ask open-ended questions, challenge the team to address issues, and ensure that everyone on the team is contributing. The moderator may estimate, but if he does, it is important that he remain unbiased by the team's estimates. A well-chosen team will allow the moderator to sit out on the estimation tasks and remain neutral and open-minded during the discussion.

The project manager should choose the team, and it should include people that she is comfortable working with. The team should include representatives from as many areas of the development team as possible: managers, developers, designers, architects, QA engineers, requirements analysts, technical writers, etc. Most importantly, each of the team members should have a stake in the plan, meaning that his goal is to establish a plan which he can agree to and live with. This allows the Delphi process to serve as an important tool for gaining the engineering team's support for the project plan, giving all involved a feeling of ownership of the estimates on which it is based.

Finally, one or more observers—selected stakeholders, users, and managers—should be encouraged to attend the meeting. The reason that the observers are important is that they

often do not understand the engineering process and what goes into building the software. Including observers is an effective way to encourage mutual trust between the team and the nontechnical people in the organization. While the observers do not directly contribute to the numerical estimates, encouraging their involvement in the meetings will increase their feeling of ownership of the final estimates that are generated by the team. When the non-engineers participate in the discussion of assumptions and see how the team arrives at estimates, they walk away with a much greater understanding of how the engineers do their work. What's more, the assumptions are almost always discussed on a level that can generally be understood by most of the nontechnical observers. Since these assumptions usually end up focused on the most problematic areas of development, the observers leave the meetings with a much clearer picture of exactly how the software will be developed.

Kickoff meeting

The goal of the kickoff meeting is to prepare the team for the estimation session. When the kickoff meeting is scheduled, each team member is given the vision and scope document and any other documentation that will help her understand the project she is estimating. The team members should read all of the material before attending the meeting.

In addition, a goal statement for the estimation session should be agreed upon by the project manager and the moderator and distributed to the team before the session. This statement should be no more than a few sentences that describe the scope of the work that is to be estimated ("Generate estimates for programming and testing the first phase of Project X").

The moderator leads the meeting, which consists of the following activities:

- The moderator explains the Wideband Delphi method to any new estimators.
- If any team member has not yet read the vision and scope document and supporting documentation, the moderator reviews it with the team. (If this happens, the meeting should be expected to take an extra half-hour to hour.)
- The moderator reviews the goal of the estimation session with the team, and checks that each team member is sufficiently knowledgeable to contribute.
- The team discusses the product being developed and brainstorms any assumptions.
- The team generates a task list consisting of 10–20 major tasks. These tasks represent the top level of the work breakdown structure—additional detail can be generated later and/or discussed in the assumptions. This high-level task list is the basis for the estimates that are going to be created.
- The team agrees on the units of estimation (days, weeks, pages, etc.).

The team must agree on the goal of the project estimation session before proceeding with the rest of the estimation process. In most cases, the goal is straightforward; however, it is possible that the team members will disagree on it. Disagreement could focus on missing requirements, on which programs or tasks are to be included, on whether or not to estimate user documentation or support requirements, on the size of the user base being supported, or other basic scope issues.

After the assumptions are discussed, the moderator leads a brainstorming session to generate the WBS. The team breaks the project down into between 10 and 20 tasks, representing all of the project activities that must be performed. Once the team is comfortable with the WBS and the assumptions, it will feel much more knowledgeable about the context in which it will be developing the software. This, in turn, will make everyone more comfortable with the team's estimates.

Individual preparation

After the kickoff meeting, the moderator writes down all of the assumptions and tasks that were generated by the team during the kickoff meeting and distributes them to the estimation team. Each team member independently generates a set of *preparation results*, a document which contains an estimate for each of the tasks, any assumptions that the team member made in order to create the estimates, and any additional tasks that should be included in the WBS but that the team missed during the kickoff meeting. (Figure 3-1 shows the format of the individual preparation results.) Each team member builds preparation results by first filling in the tasks, and then estimating the effort for each task. An estimate for each task should be added to the "Tasks to achieve goal" section of the preparation results; the "Time" column should contain the estimate for each task.

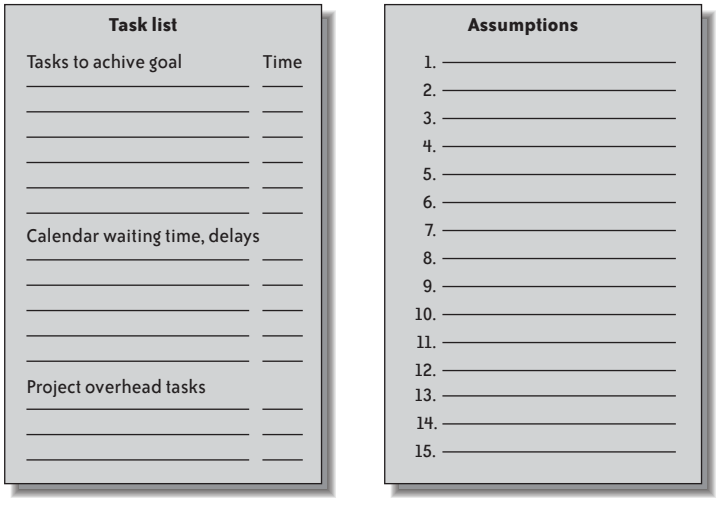


FIGURE 3-1. Individual preparation results

Each estimate should be made in terms of effort, not calendar time. This means that if the unit of estimation is "days," then the estimate should be for the total number of person-days spent. For example, if a task will require one person to work for 10 days and a second person to work for 6, the estimate should be 16 person-days (or 3.2 person-weeks, assuming a 5-day week). If both people are working at the same time so that their effort overlaps entirely, the calendar time required to do this task is 10 days.

Usually, effort does not overlap perfectly like this; this kind of parallel effort will be factored in later, when the project schedule is created (see Chapter 4). However, one important

factor in creating the schedule is taking into account necessary delays in which no work will be done. For example, the team may need to wait for a server to be built or a software licensing agreement to be reached; estimates for any known waiting time can also be added to the preparation results.

Any effort related to project overhead should not be taken into account. This includes things like status meetings, reports, vacation, etc. A separate estimation session can be held for overhead. Any time an estimator identifies a project overhead task, it should be added to the "Project overhead tasks" section of the preparation results. Similarly, estimators may run across potential delays, because certain tasks can't start until after specific dates. These should be added to the "Calendar waiting time" section, and not taken into account while making estimates. (Often, a separate Delphi session will be held specifically to estimate waiting time or overhead tasks. This is the purpose of the checkboxes at the top of the estimation form in Figure 3-2.)

Name Mike			**Date** 4/3/2004					**Estimation form** 1/1
Goal statement To estimate the time to develop prototype for customers A & B								**Units** days
Category ☑ goal tasks ☑ quality tasks ☐ waiting time ☐ project overhead								
WBS# or priority	**Task name**	**Est.**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Interview customers (A+B)	3	+2	+1				Needs off-site tri
2	Develop requirements docs	6	+5	-2	+1			Start from scratc
3	Inspect requirements docs	1	+2	+2	-2			Team of 4 BSAs
4	Do rework	1	+4					
5	Prototype design	20	-3	4	-2			Includes DB
6	Test design	5	+3					20% exists now
	Delta		+13	+5	-3			
	Total	36	49	54	51			

FIGURE 3-2. Filled-in estimation form
(©The Process Group, copied with permission)

While estimating each task, most people realize that they must make additional assumptions in order to estimate tasks. These should be recorded in the "Assumptions" section of the preparation results. They may also discover additional tasks that were not found during the kickoff meeting—these missing tasks should be added to the "Task list" section and esti-

mated along with the rest of the WBS tasks. The final result should be a complete task list, including any additional tasks found, waiting time, and overhead tasks, with an estimate attached to each task. Each team member should bring the results to the estimation session.

Estimation session

The estimation session starts with each estimator filling out an estimation form. Blank estimation forms should be handed out to meeting participants, who fill in the tasks and their initial estimates from their individual preparations. During the estimation session, the team members will use these estimation forms to modify their estimates. After the estimation session, they will serve as a record of each team member's estimates for the individual tasks, which the project manager uses when compiling the results. (Figure 3-2 shows an example of a typical filled-in estimation form.)

Before the team members fill in their forms, the moderator should lead a brief discussion of any additional tasks that were discovered during the individual preparation phase. Each task that the team agrees to add to the WBS should be added to the form; the team will generate estimates for that task later in the meeting. If the team decides that the task should not be included after all, the person who introduced it should make sure that the effort he estimated for that task is taken into account.

At this point, the participants fill out the estimation forms. The estimation form contains one row for each task being estimated. If there are more tasks than rows on the form, more than one form can be used for the session. Each participant starts with a blank form and writes the name of each task on consecutive rows. The estimate for that task should be written into the "Estimate" box next to the task. The estimate boxes are then added up and the total written into the "Total" box at the bottom of the Estimate column.

The moderator then leads the team through the estimation session:

1. The moderator collects all of the estimate forms. The estimates are tabulated on a whiteboard by plotting the totals on a line (see Figure 3-3). The forms are returned to the estimators.

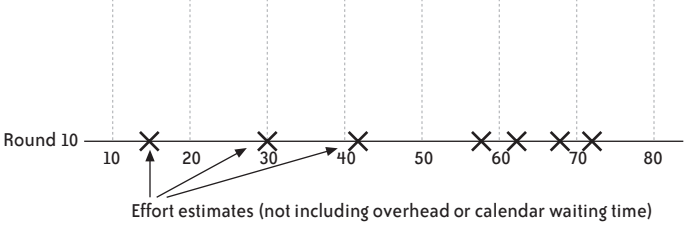


FIGURE 3-3. Initial estimates

2. Each estimator reads out clarifications and changes to the task list written on the estimation form. Any new or changed tasks, discovered assumptions, or questions are raised. Specific estimate times are *not* discussed.

3. The team resolves any issues or disagreements that are brought up. Since individual estimate times are not discussed, these disagreements are usually about the tasks themselves, and are often resolved by adding assumptions. When an issue is resolved, team members should write clarifications and changes to the task list on their estimation forms. This usually takes about 40 minutes for the first round, and 20 minutes for the following rounds.
4. The estimators all revise their individual estimates by filling in the next "Delta" column on their forms. (Using a delta allows the estimators to write "+4" or "-3" to add 4 or remove 3 from the estimate. They write the new total at the bottom of the sheet.)

This cycle repeats until either all estimators agree that the range is acceptable, the estimators feel they do not need to change their estimates, or two hours have elapsed.

Each round brings the estimates closer to convergence. Figure 3-4 shows what the typical results of an estimation session will look like. It may seem magical to the team—through a straightforward discussion of assumptions and task definition, the team arrives at a consensus that is visible on the whiteboard! This consensus is a result of clarifying these potential ambiguities. Usually the first round brings a long discussion of missed assumptions and changes to the task definition. In the later rounds, the moderator will play an important role in directing the conversation toward where it will be most effective. The moderator should look for the tasks that have the largest spread between the highest and lowest estimates. He should lead a frank discussion of the breakdown or details of that task, as well as places where the team might be over- or underestimating the effort for that task.

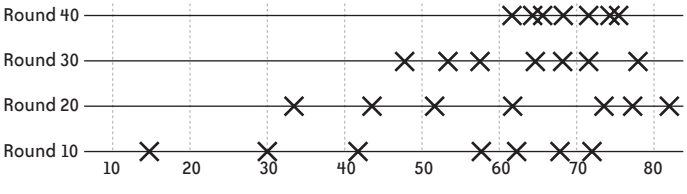


FIGURE 3-4. Converging estimate results

Disagreements often arise, but they are easier to deal with in this setting than later on, when the project is in progress. One effective way to resolve these disagreements is to talk about both sides of the issue, and then agree on an assumption that takes one of those sides. It is easier to make this decision at this stage because the assumption is not permanent; the team can very easily go back and change that assumption, if necessary. But writing down the assumption allows the team to show management that if this assumption turns out to be incorrect, the estimate may no longer be accurate. This way, even though the estimate is not perfect, the team understands why that is the case. If the team has confidence that they did a complete job with the assumptions, they will have more confidence in the value of the estimate.

After the conclusion of the estimation cycle, the moderator leads a discussion on how the session went. The team suggests ways to improve. The moderator notes their feedback, to include it in the final estimation report.

Assemble tasks

After the estimation meeting is finished, the project manager works with the moderator to gather all of the results from the individual preparation and the estimation session. The project manager removes redundancies and resolves remaining estimate differences to generate a final task list, with effort estimates attached to each task. The assumptions are then summarized and added to the list.

The final task list should be in the same format as the individual preparation results (see Figure 3-1). In addition, the project manager should create a spreadsheet that lists the final estimates that each person came up with. The spreadsheet should indicate the best-case and worst-case scenarios, and it should indicate any place that further discussion will be required. Any task with an especially wide discrepancy should be marked for further discussion. Figure 3-5 shows an example of a spreadsheet that summarizes the results. It contains a column for each estimator, as well as columns for the best-case estimate (using the lowest estimates, which assume that everything has gone well), worst-case estimate (using the highest estimates, which assume that the project hit many roadblocks), and average estimate. (When the project manager calculates these numbers, it's often a good idea to remove the highest and lowest estimates in order to eliminate outliers.)

Goal statement To estimate the time to develop prototype for customers A & B

Estimators Mike, Quentin, Jill, Sophie **Units** days

Shaded items must be discussed

WBS# or priority	Task name	M.	Q.	J.	S.	Best-case	Worst-case	Avg.-hi & lo	Notes
1	Interview customers (A+B)	6	4	3	3	3	6	3.5	
2	Develop requirements docs	5	10	2	5	2	10	5	Discrepancy between Q. and J.
3	Inspect requirements docs	7	5	6	5	5	7	5.5	
4	Do rework	8	7	9	7	7	9	7.5	
5	Prototype design	28	23	31	25	23	31	26.5	
6	Test design	9	7	6	6	6	9	6.5	
	Total	63	56	57	51	46	72	54.5	

FIGURE 3-5. Summarized results of estimation

The assumptions should also be put into a final form, suitable for inclusion in the "Assumptions" section of the vision and scope document. (If the author of the vision and scope document is not the project manager, then the project manager should turn the assumptions over to the author and help incorporate them into the document.) If the vision and scope document was already inspected and approved, it should be updated and then inspected again prior to being used as the basis for a project plan.

Sometimes there may be a team member who disagrees with the team's assessment estimate for a task, and continues to disagree with it over the course of the project. The project manager must be careful in this situation, especially if the disagreement is over a task that will be assigned to that team member. An important part of the Delphi process is that it generates consensus among the team about the final schedule. That consensus can be much harder to achieve if any of the team members feels like her objections are being ignored. One way for the project manager to make sure that this does not happen is to call an additional meeting to discuss the outlying estimates. During this meeting, the project manager must take the time to listen to everything that the person who differs from the rest of the team has to say. The final decision about the effort still lies with the project manager; however, if there is a genuine disagreement, then the person who disagrees with the rest of the team will at least feel like his objections were heard and evaluated on their merits, rather than just rejected outright.

Review results

Once the results are ready, the project manager calls a final meeting to review the estimation results with the team. The goal of the meeting is to determine whether the results of the session are sufficient for further planning. The team should determine whether the estimates make sense and if the range is acceptable. They should also examine the final task list to verify that it's complete. There may be an area that needs to be refined: for example, a task might need to be broken down into subtasks. In this case, the team may agree to hold another estimation session to break down those tasks into their own subtasks and estimate each of them. This is also a good way to handle any tasks that have a wide discrepancy between the best-case and worst-case scenarios.

Other Estimation Techniques

Wideband Delphi is not the only technique that can be effective in estimating software tasks. Here are a few popular and effective alternatives.

PROBE

Proxy Based Estimating (PROBE), is the estimation method introduced by Watts Humphrey (of the Software Engineering Institute at Carnegie Mellon University) as part of the Personal Software Process (a discipline that helps individual software engineers monitor, test, and improve their own work). PROBE is based on the idea that if an engineer is building a component similar to one he built previously, then it will take about the same effort as it did in the past.

In the PROBE method, individual engineers use a database to keep track of the size and effort of all of the work that they do, developing a history of the effort they have put into their past projects, broken into individual components. Each component in the database is assigned a type ("calculation," "data," "logic," etc.) and a size (from "very small" to "very large"). When a new project must be estimated, it is broken down into tasks that correspond to these types and sizes. A formula based on linear regression is used to calculate

the estimate for each task. Additional information on PROBE can be found in *A Discipline for Software Engineering* by Watts Humphrey (Addison Wesley, 1994).

COCOMO II

The Constructive Cost Model (COCOMO) is a software cost and schedule estimating method developed by Barry Boehm in the early 1980s. Boehm developed COCOMO empirically by running a study of 63 software development projects and statistically analyzing their results. COCOMO II was developed in the 1990s as an updated version for modern development life cycles, and it is based on a broader set of data. The COCOMO calculation incorporates 15 cost drivers, variables that must be provided as input for a model that is based on the results of those studied projects. These variables cover software, computer, personnel, and project attributes. The output of the model is a set of size and effort estimates that can be developed into a project schedule. Additional information on COCOMO can be found in *Software Cost Estimation with Cocomo II* by Barry Boehm et al. (Prentice Hall PTR, 2000).

The Planning Game

The Planning Game is the software project planning method from Extreme Programming (XP), a lightweight development methodology developed by Kent Beck in the 1990s at Chrysler. It is a method used to manage the negotiation between the engineering team ("Development") and the stakeholders ("Business"). It gains some emotional distance from the planning process by treating it as a game, where the playing pieces are "user stories" written on index cards and the goal is to assign value to stories and put them into production over time.

Unlike Delphi, PROBE, and COCOMO, the Planning Game does not require a documented description of the scope of the project to be estimated. Rather, it is a full planning process that combines estimation with identifying the scope of the project and the tasks required to complete the software.

Like much of XP, the planning process is highly iterative. The scope is established by having Development and Business work together to interactively write the stories. Then, each story is given an estimate of 1, 2, or 3 weeks. Stories that are larger than that are split up into multiple iterations. (Often, an organization will standardize on a regular story and iteration duration: for example, day-length stories and one- or two-week iterations.) Business is given an opportunity to steer the project between iterations. The estimates themselves are created by the programmers, based on the stories that are created. Finally, commitments are agreed upon. This is repeated until the next iteration of the project is planned.

Additional information on the Planning Game can be found in *Extreme Programming Explained* by Kent Beck (Addison Wesley, 2000).

Diagnosing Estimation Problems

Estimation problems almost always boil down to estimates that are either too high or too low. Padded estimates, where the team intentionally overestimates in order to give themselves

extra time, are a chronic source of estimates that are too high. Senior managers giving unrealistic, overly aggressive deadlines are a chronic source of estimates that are too low. In both cases, this can lead to morale problems.

There is a basic tug-of-war going on here. Engineers prefer higher estimates, giving them as much time and as little pressure as possible to do their work. Managers prefer to deliver things more quickly, in order to please stakeholders. The only way for a project manager to avoid this conflict is to work with the team to produce estimates that are as accurate as possible. By adopting a sound estimation process that allows the team and the project manager to reach a consensus on the effort involved in the work, the morale is maintained and the work is much more predictable.

Padded Estimates Generate Distrust

In some organizations, the project team drives the entire estimation process and the project manager simply builds a schedule around their estimates. This can be comfortable for the team, but it does not always work well for the organization, and it can eventually lead to an environment where the managers don't trust the programmers.

There are many reasons why estimates are wrong that have nothing to do with the work being done. Software engineers are often overoptimistic by nature—it's easy to be very positive about a project before doing any of the work, and it's easy to ignore problems that may come up later. It's very tempting to pad estimates, since they lead to longer schedules and less pressure.

The situation is especially bad when someone with no formal training in software engineering and little experience estimating software tasks is asked by her manager to give estimates. She is forced into a difficult situation: if her estimates come up short, she will be penalized at her next review. She could pad the estimates, but that would be dishonest. Her manager will eventually catch on and start cutting down any estimate she provides. Either of these options can lead to unreliable estimates that throw off the entire project planning process. She feels like she's left hanging with little support, and if her manager sees that her estimates are off, he could end up distrusting ones she makes in the future.

A programmer who knows that there will be ramifications—a poor review, a lower raise—if he does not meet his estimates may pad them, often extending his estimates by a factor of two or three. Managers will usually catch onto this, making it clear that they don't trust the programmer's estimates and asking for the tasks to be done early. The programmer, in turn, will only pad the estimates more. This "arms race" usually leads to a complete breakdown of the planning process.

A project manager could avoid the problem of estimates that are chronically padded by having the team reach a consensus on their estimates in an open meeting, where team members are less likely to pad their numbers. Team members who have thoroughly discussed their assumptions about the estimates are much less likely to be overly optimistic—they remain more grounded in the facts of the project, and will not sweep as many details aside. It may be tempting to pad estimates when delivering them by email; it is much

harder to pad them when sitting at a table with everyone else on the team, knowing that one's estimates might be challenged and would have to be justified on the spot.

Self-Fulfilling Prophecy

Some project managers respond to an unrealistic deadline by creating "estimates" that are too low but that meet it. Sometimes a team makes up for their project manager's poor estimates through enormous effort and overtime. When this happens, those poor estimates become a self-fulfilling prophecy—instead of being an honest assessment of the work that the team expects to do, the estimate turns into a series of baseless deadlines that the team must meet. The team never agreed to these deadlines, nor did they have any input into them; when this happens, the team will begin to resent their project manager. This is especially common when the top programmer is promoted to project manager, and fails to take into account the fact that he works faster than the rest of the team. (When this happens, the project manager's bond with other programmers will make it more likely for them to agree to the deadline and work much harder.) But any project manager is susceptible to this problem.

Typically, the self-fulfilling prophecy happens when the project manager is the sole source of estimates. He will create a project schedule with aggressive deadlines. Often, he will set the deadline first, and then work backward to fill in the tasks. The effort required to perform each task is not taken into account, nor is the relative expertise of each team member.

If the deadlines are too aggressive but not entirely impossible, the team will work to meet them. This may seem like a good thing to the project manager—he was able to get more work out of the team. But, as the team begins to burn out, they start to realize that they are working toward an unrealistic project schedule. The project does not go smoothly. Instead, it alternates between normal working time and crunch periods, and there is little advance warning before the crunch periods begin.

The first few times that the team meets an unrealistic schedule by working nights and weekends, they are happy to have met the deadline. But each time this happens, the project manager's undeservedly high opinion of his own estimating skills is reinforced. Eventually, the team gets disillusioned and bitter, and the sense of camaraderie that was forged over the many late nights and high-pressure assignments is overwhelmed by anger and frustration.

Had the project manager used a consensus-driven estimation process, the team would have been able to go into the project with a real understanding of what would be asked of them. They would know at the outset that the project would require crunch periods, and would be able to plan their lives around them instead of being blindsided by them. And the project manager would be able to keep the team together, without causing them to feel bitter or exploited.