CHAPTER TWO

Software Project Planning

AN URGENT NEED CAN BE A GOOD MOTIVATOR. Software is usually built to address an urgent need. For the stakeholders—meaning everyone who has a concern or real interest in the success of the project—the fact that the need is urgent helps to convince the organization to dedicate time and energy to addressing it. For the software engineers, an urgent need means that their work will be appreciated and used in the foreseeable future. And for the management of the organization, it means that there is a clear direction in how they need to run things.

But urgency can also be the source of a project's most difficult problems. Urgency causes people to panic, and to potentially gloss over important parts of the project. People will rush into gathering requirements (or, even worse, rush into building the code!) without thoroughly understanding the needs of the people who want the software built.

If a project manager does not really understand the context in which the software is being built, then the only thing that the project team sees is the urgency; they lose track of the needs. They can see the individual problems that they are working to solve, but they may lose track of the big picture.

Unless the team understands the needs that drive the project, they may end up with a narrow focus, causing them to waste time addressing problems that are of little importance to the stakeholders. It's easy to build great software that solves the wrong problems, but the only way to build the appropriate software is for everyone in the project to understand and agree on both why and how that software will be built before the work begins. That's the purpose of project planning.

Understand the Project Needs

When a stakeholder does not feel that his needs are being met, he usually puts pressure on the project manager to provide an early version of the software, so that he can personally verify that the team really understands why the software is being built. This is a big source of communication failure between the people who need the software and the team members who are building it. When the stakeholder asks for an early version or a prototype of the software, he is usually asking for evidence that his needs are understood and being addressed. When programmers hear this request, however, they interpret it to mean that the person is interested in the details of what they are doing, and that he wants to be given a tour of the solution as it is being developed.

This may sound funny to some seasoned software engineers. Such a basic lack of communication couldn't really be that widespread…right? But most software professionals have had to sit through uncomfortable meetings where a designer or programmer gives a demo or walkthrough of all of the great work he did, only to be surprised that nobody seems to care.

The reason nontechnical people seem so bored in demos and walkthroughs is because they came to see confirmation that the technology team understands their needs. Instead, they got a lecture in software design, which they did not want. This situation can be frustrating for everyone involved: the software engineers feel unappreciated and condescended to, while the stakeholders feel like their needs are not being taken seriously.

A project that starts out like this is likely to experience scope creep, delays, and even outright failure. What's worse, nobody in the organization will really even understand why the project had so many problems. All they will see is a team that built software that didn't work properly and had to be repaired or even rewritten. This is a very bad situation for a software engineering team to end up in. Even when a team is technically proficient and capable of delivering high-quality, well-written software, when faced with a problematic project, most managers will intuitively feel that the team is incapable of delivering software without major quality problems.

What usually stalls a project is a lack of good project management. To prevent problems, the project manager must identify the people who are making decisions that affect the project and understand why they need the software built. By talking to them and writing down their needs, the project manager can set the project on its proper course—and give the stakeholders the feeling from the very beginning that the team is taking their needs seriously.

Drive the Scope of the Project

When the project begins, the project manager has a unique role to play. The start of the project is the time when the scope of the project is defined; only the project manager is equipped to make sure that it's defined properly. Everyone else has a role to play later on: users and stakeholders will provide expertise, requirements analysts will write specifications, programmers will build the code, etc. Everyone involved in the project has some input into the scope, but only the project manager is solely dedicated to it.

Defining the scope is the most productive thing a project manager can do to get the project underway. It is usually counterproductive to do any other project activity before everybody agrees on the scope, because that activity could fall outside of the scope without anyone realizing it. For example, many programmers will immediately begin coding proof-of-concept prototypes before even talking to the users or stakeholders; many stakeholders will encourage this because they intuitively feel that they cannot make decisions without seeing something in front of them. But building a working model is a very time-consuming way to figure out that a feature is not needed. If time is a concern, this is definitely not an efficient way to build software.

By focusing on discussing the scope and writing a vision and scope document, the project manager can ensure that the team starts out moving in the right direction. This should not be hard—when the project begins, everyone is highly suggestible. Nobody is really in her area of expertise yet: the code is not ready to be built, requirements are not known well enough to be gathered, and designs and test plans cannot even be approached yet.

Instead, there is a great deal of knowledge floating around in peoples' heads. To make sure that knowledge is captured, there must be a single person responsible for gathering it. This is the main role for the project manager at the start of the project. Once he starts talking to individual people about what they expect to see in the project and writing down their thoughts, the scope will start to coalesce and people will start feeling comfortable with the direction in which it is going. This can happen very quickly once the project manager starts asking questions and writing down the answers.

This is why the project manager is the driving force at the start of the project. It may not always feel like it, but the most important project decisions are made at the project's outset. This is when the broad strokes are laid down, when the major features of the software are defined. If there is a misunderstanding at this point, the team can be sent down an entirely wrong path, which can cause them to lose all of their momentum.

When a project team is first assembled, there is almost always a sense of anticipation and excitement among the project team and the stakeholders. The project manager can take advantage of this energy to drive the project in the right direction.

Talk to the Main Stakeholder

The project manager's first task in any software project is to talk to the main stakeholder. Unfortunately, it's not always immediately obvious who that stakeholder is. The project manager should find the person who will be most impacted by the project, either because he plans on using it or because he is somehow responsible for it being developed. In other words, the project manager needs to find the person who will be in trouble if the software is not developed. (There are often several people who are in this situation. The project manager should talk to each one, starting with the person who he feels will provide the most useful information.)

When a project first starts, a project manager's job is not unlike that of a tailor fitting someone for a custom suit or dress. The tailor's customers will pay a premium for tailored clothes, rather than paying less for an outfit off the rack. But this customization also means that they will need to spend time with the tailor choosing the patterns, going through fabric swatches, taking measurements, and giving some of the precise instructions necessary to customize the clothing. The customer does not see this as a chore, but rather as a perk. By giving exact specifications, the customer can get exactly what he wants. The project manager should try to form the same sort of relationship with each stakeholder that the tailor does with his customers. He can do this by working to understand exactly what it is that the stakeholder will need from the software, and then by helping the project team to deliver software that is tailored to those needs.

Unfortunately for most project managers, the typical relationship with the stakeholder is more like the relationship between a car mechanic and his customer. The customer does not see that the mechanic is using specialized skills to fix a potentially difficult problem. He just can't use his car until the mechanic says it's fixed. He wants the fix to be as fast and cheap as possible, and he doesn't fully understand why it costs so much. What's more, he's always a little suspicious that the mechanic is doing more work than necessary or ordering a part that's too expensive. The customer never knows what's wrong until the mechanic examines the car, and the cost is always an unpleasant surprise. As far as the customer is concerned, the mechanic is simply removing an annoyance, and the customer resents having to pay any money at all for it. This is exactly how many stakeholders think about their software projects and the teams that build them.

For this reason, it's important for the project manager to take responsibility for the delivery of the project from its very beginning. Each stakeholder should feel like the project

manager is his main point of contact for any problem or issue with the project. He should feel that the project manager understands his needs, and that he will work to make sure that those needs are addressed in the software. He should be comfortable going to the project manager with any problems or changes at any point during the project's duration.

This sort of goodwill from a stakeholder can often be established in a single conversation or an initial meeting at the beginning of the project. The project manager can do this by blocking out time to meet with the stakeholder, leading him through a conversation about his specific needs, and then showing that he really understood those needs. The way the project manager shows the stakeholder that his needs are understood is by writing a vision and scope document, and then having the stakeholder review it.

Write the Vision and Scope Document

The *vision and scope* document is one of the most important tools that a project manager has; it is also one of the easiest to implement. A good vision and scope document will help a project avoid some of the costliest problems that a project can face. By writing the document and circulating it among everyone involved in the project, the project manager can ensure that each of the stakeholders and engineers shares a common understanding of the needs being addressed—and that the software must address them.

Some of the most common (and expensive) problems that a project can experience are caused by miscommunication about the basic goals of the project and about the scope of the work to be done. (The *scope* of a project usually refers to the features that will be developed and the work that will be done to implement those features. It often also includes an understanding of the features that will be excluded from the project.) By controlling the scope, the project manager can make sure that all of the software engineers' activities are directed toward building software that will fulfill the needs of the stakeholders.

The "vision" part of the vision and scope document refers to a description of the goal of the software. All software is built to fulfill needs of certain users and stakeholders. The project manager must identify those needs and write down a vision statement (a general statement describing how those needs will be filled).

Many software engineers will recognize the sinking, pit-of-the-stomach feeling when, upon seeing the software for the first time, a stakeholder or customer says, "But I needed the software to do this other thing, and I don't see that anywhere." The vision and scope document helps project managers avoid that problem by catching misunderstandings early on in the project, before they have a chance to influence the code and send the project down the wrong path.

When a project is initiated, the project manager should take the lead, talking to the stakeholders and creating a vision and scope document before the first line of code has been written. However, sometimes project managers must take over projects on which the programming has already been started. This is never an easy situation—senior managers don't usually change project managers unless the project is already in trouble. But even when a project manager has to take over a project that has already been started, it's still a

good idea to go back to the stakeholders and build a new vision and scope document. This is the project manager's best chance at catching scope problems as early as possible. If the project has started to go off track, the best way to repair the damage is to assess the current needs of the stakeholders, and then to create a plan to change the software in order to meet those needs.

Table 2-1 shows a typical outline for a vision and scope document. When talking to each stakeholder, the project manager should direct the conversation so that all of the topics in the vision and scope document are discussed. Typically, stakeholders will be happy to talk about all of this. They know their own needs very well, they understand who will be using the software, and they generally have many opinions about what features should go into it. The project manager should take lots of notes during this conversation—enough so that writing the vision and scope document is mostly a matter of transcribing those notes. (Stakeholders will often appreciate lots of note-taking, because it shows that the project manager is listening to them and taking what they say seriously.)

TABLE 2-1. Vision and scope document outline

1. Problem Statement
a. Project background
b. Stakeholders
c. Users
d. Risks
e. Assumptions
2. Vision of the Solution
a. Vision statement
b. List of features
c. Scope of phased release (optional)
d. Features that will not be developed

The outline itself provides a good guide for the discussion with each stakeholder. It can be used as an agenda for the meetings that the project manager uses to gather the information about stakeholder needs. It's a common mistake for a project manager to approach the writing of a vision and scope document as little more than a bureaucratic exercise. The project manager's goal should be to learn what each user, stakeholder, and project team member thinks about the software in order to develop a single, unified vision and ensure that everyone shares that vision. He should treat the document as a tool to build consensus among the stakeholders and project team members. In other words, the important part is the discussion of the vision and scope; the document is simply a record of that discussion.

Project background

This section contains a summary of the problem that the project will solve. It should provide a brief history of the problem and an explanation of how the organization justified the decision to build software to address it. This section should cover the reasons why the problem exists, the organization's history with this problem, any previous projects that were undertaken to try to address it, and the way that the decision to begin this project was reached.

Stakeholders

This is a bulleted list of the stakeholders. Each stakeholder may be referred to by name, title, or role ("support group manager," "CTO," "senior manager"). The needs of each stakeholder are described in a few sentences.

Users

This is a bulleted list of the users. As with the stakeholders, each user can either be referred to by name or role ("support rep," "call quality auditor," "home web site user")—however, if there are many users, it is usually inefficient to try to name each one. The needs of each user are described.

Risks

This section lists any potential risks to the project. It should be generated by a project team's brainstorming session. It could include external factors that may impact the project, or issues or problems that could potentially cause project delays or raise issues. (The process for assessing and mitigating risk below can be used to generate the risks for this section.)

Assumptions

This is the list of assumptions that the stakeholders, users, or project team have made. Often, these assumptions are generated during a Wideband Delphi estimation session (see Chapter 3). If Wideband Delphi is being used, the rest of the vision and scope document should be ready before the Delphi meeting and used as the basis for estimation. The assumptions generated during the estimation kickoff meeting should then be reviewed, and any nontechnical assumptions should be copied into this section. (Technical assumptions—meaning assumptions that affect the design and development but not the scope of the project—should not be included in this document. The estimate results will still contain a record of these assumptions, but they are not useful for this particular audience.)

If Wideband Delphi is not being used to generate the assumptions, the project manager should hold a brainstorming session with the team to come up with a list of assumptions instead. (See Chapter 3 for more information on assumptions.)

Vision statement

The goal of the vision statement is to describe what the project is expected to accomplish. It should explain what the purpose of the project is. This should be a compelling reason, a solid justification for spending time, money, and resources on the project. The best time to write the vision statement is after talking to the stakeholders and users and writing down their needs; by this time, a concrete understanding of the project should be starting to jell.

List of features

This section contains a list of features. A *feature* is as a cohesive area of the software that fulfills a specific need by providing a set of services or capabilities. Any software package—in fact, any engineered product—can be broken down into features. The project manager can choose the number of features in the vision and scope document by

changing the level of detail or granularity of each feature, and by combining multiple features into a single one. Sometimes those features are small ("screw-top cap," "holds one liter of liquid"); sometimes they are big ("four-wheel drive," "seats seven passengers"). It is useful to describe a product in about 10 features in the vision and scope document, because this usually yields a level of complexity that most people reading it are comfortable with. Adding too many features will overwhelm most readers.

Each feature should be listed in a separate paragraph or bullet point. It should be given a name, followed by a description of the functionality that it provides. This description does not need to be detailed; it can simply be a few sentences that give a general explanation of the feature. However, if there is more information that a stakeholder or project team member feels should be included, it is important to include that information. For example, it is sometimes useful to include a use case (see Chapter 6), as long as it is written in such a way that all of the stakeholders can read and understand it.

Scope of phased release (optional)

Sometimes software projects are released in phases: a version of the software with some subset of the features is released first, and a newer, more complete version is released later. This section describes the plan for a phased release, if that approach is to be taken.

This is useful when there is an important deadline for the software, but developing the entire software project by that deadline would be unrealistic. The most common way to compromise on this release date is to divide the features into two or more releases. In that case, this section should identify specifically when those versions will be released, and which features will be included in each version. It's reasonable to divide one feature up between two releases, as long as it is made clear exactly how that will happen.

If a project manager needs to release a project in phases, it is critical that the project team be consulted. Some features are much more difficult to divide than others, and the engineers might see dependencies between features that are not clear to the stakeholders and project manager. After the phased release plan is written down and agreed upon, the project team should always be asked to re-estimate the effort and a new project plan should be generated (see below). This will ensure that the phased release is feasible and compatible with the organization's priorities.

Features that will not be developed

Features are often left out of a project on purpose. When a feature is explicitly left out of the software, it should be added to this section to tell the reader that a decision was made to exclude it. For example, one way to handle an unrealistic deadline is by removing one or more features from the software, in which case the removed features should be moved into this section. The reason these features should be moved rather than deleted from the document is that otherwise, readers might assume that they were overlooked and bring them up in a review. This is especially important during the review of the document because it allows everyone to agree on the exclusion of the feature (or object to it).

Review the vision and scope document

Once the vision and scope document has been written, it should be reviewed by every stakeholder, by the members of the project team, and, ideally, by at least a few people who will actually be using the software (if they are available). Performing this review can be as simple as emailing the document around and asking for comments. The document can also be inspected (see Chapter 5). However the document is reviewed, it is important that the project manager follow up with each individual person and work to understand any issues that the reviewer brings up. The project manager should make sure that everyone agrees that the final document really reflects the needs of the stakeholders and the users, and that if they build the software described in the second half of the document, then all of the needs in the first half will be met. Once the document has been reviewed and everyone agrees that it is complete, the team is unified toward a single goal and the project can be planned.

> NOTE
>
> More information on understanding user and stakeholder needs can be found in *Managing Software Requirements: A Use Case Approach* by Dean Leffingwell and Don Widrig (Addison Wesley, 2003). More information on defining features and creating a vision and scope document can be found in *The Art of Project Management* by Scott Berkun (O'Reilly, 2005). A more detailed template for a vision and scope document is described in *Software Requirements* by Karl Wiegers (Microsoft Press, 1999).

Create the Project Plan

The *project plan* defines the work that will be done on the project and who will do it. A typical project plan consists of:

- A statement of work that describes all work products (specifications, test plans, code, defect reports, and any other product of work performed over the course of the project) that will be produced and a list of people who will perform that work
- A resource list that contains a list of all resources that will be needed for the product, and their availability
- A work breakdown structure and a set of effort estimates (described in Chapter 3)
- A project schedule (described in Chapter 4)
- A risk plan that identifies any risks that might be encountered and indicates how those risks would be handled, should they occur

The project plan is used by many people in the organization. The project manager uses it to communicate the project's status to the stakeholders and senior managers, and to plan the team's activities. The team members use it to understand the context for the work they are doing. The senior managers use it to verify that the project's cost and schedule are

reasonable and under control, and that the project is being done in an efficient and cost-effective manner. The stakeholders use it to make sure that the project is on track, and that their needs are being addressed.

It is important that the organization reach consensus on the project plan. To accomplish this, the plan should be inspected by the representatives of the project team, senior management, and stakeholders. Many project plans are stored simply as a folder containing a set of word processor and spreadsheet files, or are printed and stored in a binder. It is important that the project plan is periodically reviewed, and that any deviations from the plan are tracked at the review sessions. Frequent reviews are what can keep the plan from going stale and becoming a work of fiction.

It's difficult, if not impossible, to build a project plan without a vision and scope document. Without it, the actions that a project manager would have to take in order to create a project plan are almost identical to those required to create the vision and scope document. For this reason, the project manager should begin the planning process by first writing a vision and scope document; all other planning activities depend on it, and the time required for the project manager to create it will pay for itself when the project plan is created.

Statement of Work

The first component of the project plan is a *statement of work* (SOW). This is a detailed description of all of the work products that will be created over the course of the project, including who is responsible for creating each work product. The description of each work product should contain a reference to any tasks in the project schedule (see Chapter 4) in which it is involved. The vision and scope document is a useful starting point for the SOW. But the SOW serves a different purpose—while the vision and scope document talks about the rationale for the project (the needs that must be met, the list of users and stakeholders who need it built, etc.) the SOW simply contains a detailed list of the work that must be done and all of the work products that will be produced.

The SOW is included as part of the project plan, but it should be a separate document that can stand on its own. It should contain each of the following:

- The list of features being developed. If the software is being released in phases, the features should be divided into those phases as well.
- A description of the intermediate deliverable or work product that will be built. This is a list that covers (but is not limited to): software requirements specifications, design and architecture specifications, class or UML diagrams, code or software packages (divided into separate libraries or modules, if necessary), test plans and test cases, user acceptance plans, and any other document, source code or other work product that will be created. A brief description—no more than a paragraph—is usually sufficient for each one. The SOW also should list any standards or templates that will be used to create the work product.
- The estimated effort involved for each work product to be delivered (possibly based on the results of the Wideband Delphi estimation session), if known.

Resource List

The project plan should contain a list of all resources that will be used on the project. This list should go beyond what's covered by the project schedule by including a description of each resource, as well as any limits on that resource's availability.

Most project management software packages provide a feature to maintain a resource list. If this is not available, the resource list can either be a spreadsheet or a word processor document containing a simple list, with one line per resource. The list should give each resource a name, a brief one-line description, and the availability and cost (if applicable) of the resource. All resources should be handled in the same way, regardless of type.

Estimates and Project Schedule

Once the statement of work and the resource list have been created, the project manager should build a project schedule. This is usually done in several steps:

- A work breakdown structure (WBS) is defined. This is a list of tasks that, if performed, will generate all of the work products needed to build the software.
- An estimate of the effort required for each task in the WBS is generated.
- A project schedule is created by assigning resources and determining the calendar time required for each task.

The project plan should include the complete revision history of the WBS—it should contain a list of any tasks that are added, changed, or removed, and when those changes occurred. It should also include estimates and a project schedule, including any revisions that were made during the review meetings. (Chapter 3 contains a repeatable process for generating a WBS and estimates. Chapter 4 describes how to create a project schedule.)

Risk Plan

A *risk plan* is a list of all risks that threaten the project, along with a plan to mitigate some or all of those risks. Some people say that uncertainty is the enemy of planning. If there were no uncertainty, then every project plan would be accurate and every project would go off without a hitch. Unfortunately, real life intervenes, usually at the most inconvenient times. The risk plan is an insurance policy against uncertainty.

Each of the risks in the plan must be assessed by the project manager and the team. Risk assessment is an important part of planning a software project because it allows the project manager to predict potential problems that will threaten the project and take steps to mitigate those problems. Adding a risk plan to a software project plan is an effective way to keep the project from being derailed by surprises or emergencies. Many people are thrown off by the word "assessment"—it's a word that is usually associated with finances or accounting, not with project management. But in this case, it's appropriate. A good project manager will assess the risk of her projects in much the same way that a good stock trader will assess the risk of his portfolio. In both cases, potential problems should be identified, and the relative probability and impact of each risk should be estimated. Certain risks will

be much more likely to occur and, if they do occur, they might cause much more damage than others. In those cases, steps should be taken to hedge the project (or portfolio) against the risk; this is usually referred to as "mitigation" when it is done in the context of project planning.

Risk planning for most projects can be done in one meeting, usually in less than two hours. The meeting is led by the project manager, who should select a team similar (or identical) to that of the Wideband Delphi session (see Chapter 3), with the exception that there is no moderator. Table 2-2 contains a script for creating the risk plan.

TABLE 2-2. Risk planning script

Name	**Risk planning script**
Purpose	To assess risks and create a risk plan.
Summary	The risk planning meeting happens in three parts: a brainstorming session to identify risks; a discussion in which the probability and impact of each risk is estimated; and a discussion to identify actions that can mitigate risks. The end result is a risk management plan, which should be included verbatim in the final project plan.
Work Products	*Input* Any project documentation that has been developed so far. *Output* Risk plan. Assumptions generated by the Delphi process. Assumptions in the vision and scope document.
Entry Criteria	The project manager has gathered the project team for a two-hour meeting to assess the project's risks.
Basic Course of Events	1. *Brainstorm potential risks.* The project manager leads a brainstorming session to identify risks. Team members suggest every risk they can think of; the project manager writes the risks on a whiteboard as they come up. Brainstorming should be reminiscent of microwave popcorn: a few ideas should "pop" at first, followed by a large number being fired rapidly, slowing down to a final few "pops." The team will generally be able to judge when the risk identification is over. 2. *Estimate the impact of each risk.* The team assigns a number from 1 (highly unlikely) to 5 (very likely to occur) to represent the estimated probability of each risk. Similarly, impact should be estimated by assigning a number from 1 (for a risk with low impact) to 5 (for a risk which, if it occurs, will require an enormous effort to clean up). 3. *Build the risk plan.* The team identifies actions to be taken to mitigate high-priority risks and creates a risk plan that documents these actions.
Exit Criteria	The risk plan is finished.

Brainstorm potential risks

Risks should be as specific as possible. It's true that "The project might be delayed" or "We will go out of business" are risks; however, they are far too vague to do anything about. When a vague risk comes up, the project manager should prod the team into making it more specific. What are the possible sources of the project delay? How have past projects been delayed? For example, if the last project was late because a major stakeholder quit and was replaced by someone who disagreed with his predecessor's vision, the team should write that down as a risk. The assumptions documented in the vision and scope

document and identified in a Delphi session are another good source of potential risks. The team should go through them and evaluate each assumption for potential risks as part of the risk brainstorming session.

Estimate the impact of each risk

Once the team has generated a final set of risks, they have enough information to estimate two things: a rough estimate of the probability that the risk will occur, and the potential impact of that risk on the project if it does eventually materialize. The risks must then be prioritized in two ways: in order of probability, and in order of impact. Both the probability and impact are measured using a relative scale by assigning each a number between 1 and 5.

These numbers are arbitrary; they simply are used to compare the probability or impact of one risk with another, and do not carry any specific meaning. The numbers for probability and impact are assigned to each risk; a priority can then be calculated by multiplying these numbers together. It is equally effective to assign a percentage as a probability (i.e., a risk is 80% likely to occur) and a real duration for impact (i.e., it will cost 32 man-hours if the risk occurs). However, many teams have trouble estimating these numbers, and find it easier to just assign an arbitrary value for comparison.

Many people have difficulty prioritizing, but there is a simple technique that makes it much easier. While it's difficult to rank all of the risks in the list at once, it is usually not hard to pick out the one that's most likely to occur. Assign that one a probability of 5. Then select the one that's least likely to occur and assign that one a probability of 1. With those chosen, it's much easier to rank the others relative to them. It might help to find another 5 and another 1, or if those don't exist, find a 4 and a 2. The rest of the probabilities should start to fall in place. Once that's done, the same can be done for the impact.

After the probability and impact of each risk have been estimated, the team can calculate the priority of each risk by multiplying its probability by its impact. This ensures that the highest priority is assigned to those risks that have both a high probability and impact, followed by either high-probability risks with a low impact or low-probability risks with a high impact. This is generally the order in which a good project manager will want to try to deal with them: it allows the most serious risks to rise to the top of the list.

Make a mitigation plan

All of this risk brainstorming and estimation is only useful if it leads to the team taking actions to avoid the most pressing risks. The remainder of the risk planning meeting should be dedicated to identifying these actions. The project manager should start with the highest-priority risk, working with the team to decide on any actions that should be taken. After that, the team should move down the list of risks, until they decide that the priority of each of the remaining risks is low enough that no action would be required.

The team can take any or all of these actions to mitigate a risk:

- *Alter the project plan.* The project schedule can be adjusted to help reduce the risk. Riskier tasks can be moved earlier in the project, or given more time. This will give the team an early warning or a time cushion in case the risks materialize. The project manager can also hold an additional estimation session to break down the riskiest tasks into subtasks. More detailed planning will help reduce the risk.
- *Add additional tasks.* There are certain actions that can be added to the schedule to help avoid risks. For example, if there is a high probability that a critical team member will leave the organization, cross-training tasks can be assigned to other people. This will increase total effort in the project, but it will be worth it if the team member leaves.
- *Plan for risks.* For risks with a high impact that do not need specific tasks or project plan changes, the project manager should have the team spend a few minutes identifying the steps that should be taken in case the risk does occur. These do not need to be added to the project schedule, but they should be written down and added to the risk plan. This way, if the risk does occur, nobody will panic. Problems that have a large impact on the project can be demoralizing to the team and may throw the project into chaos. Simply having preplanned the steps needed to fix the problem is highly reassuring; it keeps the team feeling like they are on track.

Once the mitigation steps are identified, all of these risks and actions should be documented in a risk plan. The easiest way to do that is to create a simple spreadsheet with five columns: Risk (one to three sentences that describe each risk), Probability (the estimated probability from 1 to 5), Impact (the estimated impact from 1 to 5), Priority (Probability × Impact), and Action (the specific actions that will be taken to mitigate the risk, or "None" if the risk is deemed a low enough priority to ignore). Figure 2-1 shows a sample risk plan.

> **NOTE**
> **A more detailed risk mitigation process is described in *Making Process Improvement Work* by Neil Potter and Mary Sakry (Addison Wesley, 2002).**

Project Plan Inspection Checklist

The project plan—including the project schedule—should be reviewed (see Chapter 5) using this inspection checklist:

Statement of work

Does the project plan include a statement of work (SOW)?

Is the SOW complete—does it contain all of the features that will be developed?

Are all work products represented?

If estimates are known, have they been included?

Resources

Does the project plan include a resource list?

Does the resource list contain all resources available to the project?

Risk plan for project	Call center application project			
Assessment team members	Mike, Barbara, Quentin, Jill, Sophie, Dean, Kyle			
Risk	**Prob.**	**Impact**	**Priority**	**Actions**
Senior management will move call center offshore, which will require an internationalization feature to be built	3	5	15	1. Mike will add a requirements task to the schedule for Quentin to begin investigating internationalization requirements. 2. If the call cener is moved, Mike will call a team meeting to review the schedule and Barbara will inform the rest of senior management of the potential delay.
Jim will be pulled off of this project for Royalty Archive project bug fixes	4	3	12	1. Assign Kyle to work with Jill on the initial programming tasks to make sure he is cross-trained. 2. If Jill is pulled off, she will spend 10% of her time reviewing this project with Kyle.
Reporting feature will be needed	2	4	8	If this happens, Mike will work with Sophie and Kyle to reestimate the programming tasks.
Additional time will be needed to gather requirements from potential users at Boston client	5	1	5	None
Will need to support tie-in to support additional database vendors	1	3	3	None

FIGURE 2-1. Sample risk plan

Are there any resources known to be assigned to other projects at the same time that they are assigned to this one?

Have dates that the resources are unavailable (scheduled downtime for machines, vacations for people, times that facilities cannot be booked, etc.) been taken into account?

Project schedule

Does the project plan include a schedule?

Are there any tasks that are missing or incorrect?

If a WBS was generated by a Delphi session, does the project schedule reflect all of the tasks that were identified by the team?

Does each task have a predecessor?

Is a resource allocated to each task?

If multiple resources have been assigned to a single task, has the task's duration been updated properly to reflect that?

Is there a more efficient way to allocate resources?

Does the project schedule contain periodic reviews?

Risk plan

Does the project plan include a risk plan?

Are there any risks that are not in the plan?

Are there any assumptions (from the vision and scope document or a Delphi session) that represent risks that should be included in the plan?

Is each risk prioritized correctly?

Has the impact of each risk been estimated correctly?

Have the risks been sufficiently mitigated?

Diagnosing Project Planning Problems

If a project is not planned well, it will veer off course fairly quickly. If the project manager does not take the lead in defining the scope immediately, the project will quickly become chaotic. Even if the scope seems to be defined well, the project manager must make sure that all stakeholders really understand and agree to it in order to avoid problems later on in the project. The team must buy into the scope as well, or else they will make decisions that are not in line with the project goals.

Lack of Leadership

It's not uncommon for people to intuitively feel that all they need for a project to be successful is a group of highly talented and motivated people. But even the best people will have trouble starting a project if nobody takes the lead.

One common problem that comes from a lack of leadership is tunnel vision. Each team member knows how to do her specific tasks; however, there's no way to plan for every detail of that task. She will almost certainly encounter decision points. For example, she may see a better way to solve a particular problem that will cost time but lead to a better solution. For some projects, it may be appropriate to pursue this; for others, the delivery schedule is more important than the superior solution.

Without good leadership, the team member might be afraid to make this decision. This usually results in the team member sending emails to peers, managers, stakeholders, and anyone else she can find, requesting confirmation of absolutely every little decision that gets made. People quickly get inundated with notes about project details that they lack the context to even understand.

On the other hand, she may also simply make decisions based on her gut feelings. As the project progresses, not all of her decisions are in line with the needs of the stakeholders. For example, she may choose to pursue the superior technical solution at the expense of the deadline, despite the fact that the stakeholders' needs would have been just as easily satisfied had she chosen the faster solution. As these decisions pile up throughout the course of the project, the scope itself starts to drift away from the organization's needs.

If there is a serious enough lack of leadership, the project may not even get this far. If the scope is never fully defined, then the project may have several false starts. Designers and programmers start building prototypes to demonstrate to stakeholders, only to find that they have to go back and rebuild them because they misunderstood the project needs. Work on the code may begin, only to have programmers moved off of the project for higher priority tasks. People are given conflicting priorities and do not know which one to work on first. The project takes a long time to emerge from the chaos. Even if a product is