**Chapter 10. Management and Leadership**

**Many management books** sell themselves as lists of "best practices" that, if followed, will yield projects that are planned and executed smoothly and without any problems. Most people who try to follow those practices find that it is much harder to do in practice than the books led them to believe. Projects are not always predictable. The organization's needs may change; people may quit or be transferred into or out of the team; or the goals of the project or the climate in which the organization does its business may change. A project manager usually cannot control any of these things.

The tools and techniques in this book will help solve the most common problems that plague software projects. But there are many other ways that a project can go wrong, and it is impossible to prepare in advance for all of them. It is up to you, the project manager, to be smart. You should use these tools when you can. But you will undoubtedly come across issues or problems that these practices simply do not address, and it is your job to think your way through the solution. If you keep in mind some sound engineering principles and fundamental ideas about management, you stand a better chance of leading your projects through these problems and, in the end, delivering better software.

It is also part of your job as a manager and a leader to adequately explain the decisions that you are making, and to keep the team's interest in line with the organization's interests—and vice versa. You must do this by working with senior management to understand their goals and needs, and helping them understand that the changes you are making will help them achieve those goals. You must also work with each team member to understand her goals and needs, and help her understand the job that she must perform. If you do this, you will ensure that the software is mutually beneficial for both the organization and the individuals building it.

In a sense, part of the job of the project manager is to serve as an information conduit. You help information flow from the team up to senior management in the form of project status and analysis information. It is your job to understand all of the work being done, so that it can be summarized to the people who make the decisions about the future of the project; they need this information to make informed and intelligent decisions. This requires that the project manager put a lot of effort into understanding what it is the team is doing and why they are doing it. The project manager cannot simply ask for estimates, fit those estimates in a schedule, and quiz the team on the percentage they've completed. He must understand what actions each team member is taking to complete the task, and what possible complications they are running into. The project manager is the only person looking at how the tasks interrelate; he is the only one with the perspective to see the problems and, ideally, fix them.

**Take Responsibility**

The world is full of frustrated project managers. Frustrations come in many forms. Some project managers are assigned projects, but have to fight for the people to do those projects. Others have inadequate office space, computers, or networks. Many project managers are constantly clashing with other managers because of "dashed-line" organization structures where their teams do not report directly to them. Some find that their project team members are routinely pulled off of their projects without warning, or that their projects are reprioritized and thrown into disarray.

There is a common root cause of most, or all, of these problems. The cause is that the project manager is told that she has responsibility for her projects, but, though she is held accountable for their success, she is not given sufficient authority to do her job.

Managers routinely throw around the word "responsibility," often in the context of a subordinate "not taking responsibility" for a task. To many of them, "take responsibility for this task" is synonymous with "go away and don't bother me until the task is complete." This is an unfortunate attitude, and it is a root cause of failed projects and depleted morale in organizations around the world. A good project manager must have a more sophisticated understanding of responsibility in order to avoid these problems.

A person has *responsibility* for a task only if he is given sufficient authority to perform the task and is held accountable for the results. When you assign a task to your project team, you must ensure that each team member has sufficient authority to perform the task, as well as an understanding of his or her accountability. For the project to be most effective, the team members should understand these concepts as well.

**Ensure That You Have Authority to Do the Project**

A person has *authority* to perform a task only if he is has adequate control over the resources necessary to complete the task. Giving a project manager authority to carry out a project means giving him control over the resources (people, office space, hardware, software, etc.) required to complete it. Since resources cost money, sufficient budget for the project must be allocated within the organization.

This does not mean that the project manager must have direct control over these resources. For example, the team members do not necessarily need to report to him. However, if they do not report to him, he must have the full cooperation of the direct manager of each resource assigned to the project, so that he can assign tasks to the team members directly without having to obtain permission for each task. If there is a single person on the project whose involvement is not guaranteed from the outset, the project manager cannot say with certainty that he has all of the resources that he needs to complete the project. In this case, he does not have sufficient authority to do his job.

In the same way, the project manager does not need to have a corporate credit card to buy the necessary hardware or software. But he does need to have a guarantee from the person who has the budgetary authority that he will be allowed to obtain these resources. Without this guarantee, his authority will still be incomplete.

**You Are Accountable for the Project's Success**

A person is *accountable* for a task if failure to adequately perform that task carries professional consequences. These professional consequences can take one or more of four possible forms:

* His reputation is damaged among his peers and in the organization.
* His manager gives him a poor performance review.
* His compensation is reduced.
* His responsibilities are changed (or taken away altogether, if he is fired).

When someone is held accountable for a task, he must have some understanding of the professional consequences if he fails. If there are no consequences, or those consequences are not particularly damaging, then there is little incentive to complete the task.

That does not mean that someone who does not have incentive to complete a task will do a poor job. She may still perform it well, usually out of a sense of duty, loyalty, or personal responsibility. But if failure does not carry consequences, she is not really accountable. She is doing the task as a favor, and any success is coincidental.

**Grant Authority and Accountability to Team Members**

When you are responsible for a software project, you are accountable for its success. However, you are not the only person accountable—you must distribute that accountability fairly among the project team. The way to do that is to make sure each team member is responsible for her task by ensuring that she has sufficient authority to carry it out and understands how she will be held accountable for its completion.

The best way to ensure that each team member has sufficient authority is to discuss it directly at team meetings. When you discuss the status of a project, verify that each person has everything necessary to perform her assigned tasks.

The most common way that authority is removed is when a team member is pulled off of the task. If you have been given a person's time for your project, you must be able to depend on that to last for the course of the project. Many people will not tell you that they have additional demands on their time unless you ask them directly.

Never assign a task to a person who does not have the authority to perform it. All engineers must have control over their time. Requirements analysts must be allowed to call meetings with stakeholders. User interface designers must be allowed to make UI design decisions without having to clear them with a string of senior managers. Programmers must be allowed to use the tools and techniques that they need. Testers must be allowed to request requirements specifications, technical specifications, and preliminary builds, and must feel free to report defects without being blamed.

If the project is late or runs into problems, you must give every project team member your guarantee that you will work hard to identify the root cause of the problem. For example, many delays that are introduced due to poor planning or scope creep are not recognized until late in the project, during the testing activities. The testers may have done their jobs and met all of their estimates, but, since they are in charge of the active task at the time that the project delay is discovered, they are held accountable for problems that they had no authority to prevent. It is your job to prevent this from happening by holding the people responsible for the delay accountable. You must share in the consequences because you failed to recognize the problem until too late. And you must take steps to prevent it in future projects, by implementing additional tools and techniques.

**Defend Your Project Against Challenges to Your Authority**

If resources are pulled off of your project, your authority is being challenged. You only have authority to do a task if you can command the resources necessary to complete it, and, when people are pulled off of your project, those resources are no longer available to you. However, the accountability is still in place. If you want to avoid being held accountable for the project's failure, you must recognize the challenge to your authority and defend your project against it. This is difficult, and often requires patience and negotiation.

For example, it is very common for programmers to be interrupted "for just an hour or two." If a senior manager or executive needs help with something small—say, he's having trouble using a program that he knows the programmer wrote a few months ago—he will often approach the programmer directly, without your knowledge. This puts the programmer in a difficult position, because she does not feel like she can tell this senior person that she is too busy to help. She will want to be helpful, and may simply take on the extra work without mentioning it to you.

If something like this happens, it is your job to address the situation. Though it seems innocuous, this is an indirect (though probably unintended) challenge to your authority, because a resource has been pulled off of your project. It is also a challenge to the programmer's authority, because she no longer has enough time to do her task. This is a difficult situation to fix. You might approach the senior manager directly and explain that this will cause a delay in the project, and that both you and your programmer will be held accountable for her delay. If the senior manager balks, you might ask him to share the accountability by writing an email to the project's stakeholders explaining why the project will be delayed, so that you won't get blamed.

You may have to be creative in how you solve this problem, because this can be a politically charged situation. Approach it with a cool head. Do not accuse people of trying to interfere with your project. Remember that the person you are talking to is just trying to do his job, and he probably did not realize that he was putting your project at risk. Try to talk about how this affects you and your project team, and be very specific about the consequences. Give examples of how a delay in the programmer's task will ripple down the project and cause additional delays. Most importantly, make sure that you have your facts straight before you meet with the other manager, and make sure that your own manager knows everything you are about to say, and approves.

If your manager does not back you up in defending your authority, your hands are tied. You will either have to renegotiate the schedule or accept the consequences. This is hard for many people to deal with. The best way to prepare is to make sure that you have plenty of documentation, so that when you are called upon to explain the delay, you can show what caused it and, if necessary, that your manager knew about it and did not let you fix the situation.

# Do Everything Out in the Open

It is a very frustrating situation for a project manager when he makes a decision or takes an action, and then finds that a colleague or senior manager disagrees with him for seemingly no reason at all. Many times, these disagreements arise from a lack of communication . If everyone in your organization has constant access to everything that you and your team produce, the mystery behind why you make your decisions goes away. People may still disagree with you, but at least they will be disagreeing with your ideas and not simply because they felt like they were kept out of the loop.

It's not possible to tell everyone everything all of the time. But if it is known to the organization that a project manager is sharing his project information, and all of his colleagues know where they can find that information, they are much less likely to feel like information is being hidden from them.

Whether you are interacting with your team or with your organization's management, it is important that everything that you do is transparent. This means that when you create a document, hold a meeting of interest to others, or make an important project decision, you should share all of the information produced and used with everyone involved.

## Publish Your Work Products

All work products should be kept in a public repository. This could be a shared folder or directory, a version control system, a Wiki or other sort of web interface, a knowledge base, or some other system for information storage. This ensures transparency for both team members and the organization's management.

When each person on the team knows that the work she is doing can be read and used by all other team members, she will feel much more accountable for her work than if she were doing the same thing in private. In general, people tend to create more readable documents, build more maintainable designs, and write more readable code if they know it will be shared with others.

Managers also benefit from transparency of work products. For example, if a product ships and a client encounters a defect, a client support manager can consult the test plan for the part of the product in which the defect was found. He might want to see the defects reported in the defect tracking system, or check the specification for the feature to verify that it is indeed a defect and not a misunderstood feature. Publishing the work products allows everyone in the organization to use them as reference materials.

Your senior managers will especially benefit from transparency. If it is your responsibility to specifically summarize and report every aspect of every product, it is highly likely that you will, on occasion, leave out an important item. However, if your boss is used to looking at the project documents himself, there is no chance that you will leave him in the dark. This will help build trust between you, and will also help discover any impending problems.

Another problem that is avoided through work product transparency is information hoarding. Sometimes an insecure person feels that he needs to keep certain aspects of his day-to-day work secret from the rest of his organization—even his manager. This helps him feel more important to the organization, since any time anyone needs access to that information, they must go through him. In some cases, it even (unfairly) provides job security: if he's the only person who has maintained that particular work product, it is much harder to fire him if he's doing a poor job. That secrecy also makes it difficult to judge how poor a job he's doing.

## Make Decisions Based on Known Guidelines

If you do things the same way every time, the people who work with you will come to understand the reasoning behind your decisions. They will feel much more comfortable with you than if you make decisions in a less predictable manner. One way to help others understand your perspective, and avoid surprising them, is to publish the standards by which you manage.

There are several ways that guidelines can help make your decisions more predictable:

* Use published standards documents to help others understand the way certain roles must be filled. For example, a standard for interacting with a version control system might require that each programmer verify that the code builds without compilation errors before it is checked in. Programming standards may include naming conventions for variables or files. A testing standard might specify that a test plan must be executed by somebody other than the person who wrote it. Acceptance criteria and release readiness criteria are useful standards to help the organization make unbiased and objective decisions about when to release the software into test or to the general public. In addition, inspection checklists are also a kind of standards document.
* Documents should be based on templates when possible. This ensures that all of the information that is needed in the document is included, and that important omissions are noted by the person writing the document. For example, a template might require that a vision and scope document always have a section for future releases. For projects that are only expected to have a single release, this section will contain "N/A" or a placeholder. This will prevent the reader from wondering whether the author meant that there would be a single release, or whether it was an oversight.
* Process documents ensure that each project is done using a repeatable process. That ensures that the same activities for the current project are done in the same order as previous projects. This helps each person on the project understand how their work fits into the big picture, and reassures them that they are doing the right tasks. Process documents also give them the ability to compare the team's performance from project to project, to determine whether the organization is improving over time. The scripts used throughout this book are examples of process documents.
* Use performance plans to set expectations for individual team members. Each person in the organization should have her performance measured against a written standard, and she should be given an active role in helping to define that standard. This helps her gauge how she is performing and provides a positive environment for her manager to help fix performance problems and reward good work.

# Manage the Organization

An important part of the project manager's job is managing upward in the organization. The way that you interact with your organization's senior management can make or break your projects. When you make changes for the better, you are changing their organization, and, whether or not you are successful, your boss will want to be involved.

## Senior Managers See Software Projects as a Cost Burden

Many project managers face an uphill battle when interacting with their organizations' senior management. They find that senior managers have an increasingly antagonistic view of their software projects. The senior managers only see the cost of the development, and often fail to see how the software projects help the organization. These problems are compounded when projects come in late, or do not fill the needs of the stakeholders.

In the mid 1990s, Mary Lacity and Rudy Hirscheim published a study of 14 Fortune 500 companies in which they interviewed over 60 senior managers about their attitudes toward IT projects. They found that the overwhelming majority of them thought of their IT departments as a "cost burden" that steadily increases their costs without adding to the profitability of the company. This attitude is pervasive not just in large companies, but in organizations of all types and sizes.

Unfortunately, most project managers cannot just sit down with their organization's senior managers and explain the value of their projects. They must show over time that there is a real reason that each project is developed, and that each project's benefits justify the cost of development.

Many of the project management practices in [Part I](https://learning.oreilly.com/library/view/applied-software-project/0596009488/pt01.html) are aimed at communicating this. The vision and scope document is the project manager's first opportunity to ensure that each project is developed based on real and specific needs, and that the features of the software are aimed at fulfilling those needs. The vision and scope document and the project plan communicate the real costs and benefits of development to senior managers in terms that they understand. And a change control process ensures that the managers are kept apprised of all changes to the project, and that those changes are worth their costs.

A project is successful if its costs are justified by its benefits. Establishing a track record of successful projects is the most effective way for a project manager to reverse dangerous attitudes in senior management.

## Show Senior Managers the Impact of Their Decisions

The first step in working with senior managers is to know what a best-case scenario looks like. In good organizations, decisions are based on objective standards and metrics that were developed in advance to determine the health of the application. The goals of the project are decided from the outset, and a successful project will have met those goals. Project decisions—approval of schedules, deadlines, budgets, and resources—are made by a single person or a group of people who make those decisions based on objective evidence.

Unfortunately, in small- to mid-sized companies, this is not usually the case. Decisions are frequently made based on gut feelings instead of objective analysis. The people in charge of project decisions are not necessarily experienced in working with software projects—in many cases, this may be the first time they have encountered software projects. Even in software companies, a small or young company may be run by a manager or management team with industry, business, or organizational experience—but without much experience managing software projects. While gut decisions can be successful, they will often lead to serious project problems.

It's true that project decisions based on gut instincts are often correct. If that were not the case, people would never get into the habit of making gut decisions in the first place. But the fact is, most small businesses are run entirely on gut instincts. And if you have a small product with a small and well-understood user base, those gut decisions make intuitive sense to anyone who understands both the product and the clients. That usually includes the upper management of such a company. As a result, many small businesses have successfully built and sold software products using mostly gut instincts to govern their decisions.

One typical example of project management by gut instinct is release readiness. The typical reasoning sounds like this: "We've tested the product longer than we did for the last release, and we have not encountered any major problems. So let's release it!" In contrast, a release readiness process that was based on objective facts would require that a certain percentage of the code base is covered by the tests, and that the number of defects that are discovered falls below a certain threshold (based on the size of the application). For example, the product might only be considered releasable if at least 70% of the code has been executed under test, if no critical defects are found, and if there are fewer than 3 medium-priority and 10 low-priority defects per 10,000 lines of code. Keep in mind that the exact same testing activities could be performed in both cases; it's just that in the second case, everyone has already agreed that if these objective and measurable criteria are not met, more testing is needed.

In a small company, the senior managers are happy if they are making more money this year than last year. Most of the time, they don't try to figure out why that happened. They know that what they are doing is working, and there's no need to question it. This attitude is very common, and you must recognize its relevance to your projects.

Some senior managers feel that they have navigated more difficult problems than whether or not to put the code under source control or add another round of testing. These things seem like details to them, and they don't feel that they need to read a book to figure them out. To someone with this attitude, the very fact that you are concerned about these "details" makes you seem like an alarmist—you are up in arms about something that they consider very minor. Or, even worse, they feel that they have done a fine job of building a software organization, and are insulted that you think that you can make it better by introducing changes.

If an organization is in the software business, the people running that organization need to understand the details of making software. Making those decisions is uncomfortable for a senior manager with little software experience: it is hard to make sound decisions about building software when one has only a simple, high-level view of software design, programming, and testing.

One common senior-management response to this problem is to attempt to delegate—probably to you, since you have read a book about it and seem to know what you are talking about. Unfortunately, this is a shallow commitment that can cause its own problems. If your boss does not understand the goals of the improvements you want to make or the reasons behind them, he will not make consistent decisions. The actions that you take in implementing new practices on your project will impact other people: they will create additional work for some of them, and many will perceive that you are taking away some of their power, flexibility, or freedom. And, in some poorly run software organizations, there may be people assigned to software tasks who are not very good at those tasks; they might prefer not to have their work measured or analyzed. If they see you as an interloper, they will complain to your boss. If he does not understand why you are doing what you are doing, all he will see is conflict—conflict that you created with the changes that you made. Since he did not take the time to understand the benefits of your actions, he will simply blame you for making changes and causing conflict. Your improvement effort will grind to a halt.

The solution—and it is not an easy one—is that upper management must be better educated about the details of your project. The person who has the authority to tell you (and everyone else on your level) to undertake a project should understand the purpose behind every step in the software process. This means more than just understanding that software needs to be designed, developed, and tested. He needs to know why the software is being developed that way.

It means that if, for example, you are trying to implement inspections, then the senior manager must understand what an inspection is, what is being inspected, what kinds of defects it will find, why those defects are there, who needs to attend the inspection meetings, and what will happen if the inspection is skipped—and he needs to understand this before the inspections are implemented. He must be sold on the idea. If he does not agree that this is the most efficient and effective way to develop software, he will fold the first time someone decides that inspections are "bureaucratic," unnecessary, or somehow getting in her way. It is only with some genuine understanding of the changes you want to make that he will be able to make a real commitment to them.

The process of educating your organization's senior management must go both ways. You need to put time and effort into understanding the goals and needs of the organization and its senior managers. The easiest way to do this is to call a meeting with them. The purpose of this meeting is to write down their goals and needs. These should be written in their language (e.g., improving profits, increasing the customer base, reducing support calls, etc.). Be sure to meet with them periodically in order to keep this list up-to-date. You should be able to use it to show how your improvements will help them meet these goals.

## Don't Confuse Flexibility with Always Saying Yes

There are many situations when a project manager disagrees with the people around her. Sometimes a project is going off track or experiencing problems. Other times, people disagree with her approach to a project. It is important to be flexible in these situations. But sometimes it is hard to figure out just what it means to be flexible. Flexibility might mean making sure that everyone understands that the project is in a difficult situation, and agrees on the course that it will take. But it might also mean listening to the dissenting opinions and making changes to the way the project is being managed. Flexibility should never mean having to cave in to unreasonable requests.

### DON'T AGREE TO AN UNREALISTIC SCHEDULE

When somebody asks you to do something, it is natural to want to satisfy them—especially if that person is above you in your organization. Many project managers operate in a climate of constant pressure from above, and they want to alleviate that pressure by being positive and agreeable.

Some senior managers think that all dates are negotiable, and that teams can always be pressured into releasing software earlier—even if the team's projected date is based on realistic estimates and solid project planning. Sometimes upper managers will just challenge the team's opinion because they believe that anything can be done sooner if the team works harder. Unfortunately, just increasing the pressure on the team is a poor motivator. It makes people feel as though their opinions have been second-guessed, that their expertise is not valued, and that their work is not respected. Nevertheless, it's a common situation.

For example, consider a project in which the senior manager in charge has set a deadline, but it is clear to the project manager that this deadline is too aggressive and the team will never meet it. If she simply goes to the manager and tells him that the project is headed for failure, he will probably just think that she is not being a "team player". He will probably put a great deal of pressure on her to just accept his deadline and work the team harder, possibly forcing them to put in overtime in order to meet the goal. He would consider that being "flexible." But in reality, that's a recipe for team demotivation, and probably disaster. The senior manager's "solution" does not solve the problems that are keeping the project from delivering at that time. In fact, simply agreeing to the deadline is actually being inflexible, in that it ignores the reality of the situation and fails to present alternatives that might actually help the team meet the deadline (or, at least, satisfy the organization's needs that motivated the decision to impose the deadline).

Creating transparency and gathering consensus are the most effective ways to address this. When a project slips, the project manager needs to diagnose where the planning went wrong or what risk was not previously considered. Hopefully, she has done enough planning (see [Chapter 2](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch02.html)) that she has real evidence that the project will not meet the deadline. In this case, true flexibility would involve presenting the senior manager with options, such as doing a phased release, scaling back the features to be developed, or adding resources.

### CHANGE YOUR APPROACH WHEN NECESSARY

Sometimes a project manager is faced with a project that is coming in early. The schedule was the result of a difficult negotiation, and moving the date earlier means giving up the hard-won project time. It seems counterintuitive to move the deadline up and give up the extra padding. But there are cases when this is necessary.

One of the most common situations that results in decreasing the schedule is when a programmer discovers a shortcut. For example, a project may have gone through a solid estimation process, and a project plan was created when one of the assumptions for the estimates was that a certain component would have to be built from the ground up. For example, if a programmer discovers that this component can be replaced with one purchased off the shelf, this reduces the effort required to build the software.

Many project managers would happily sweep this under the rug and keep that extra schedule time as a buffer against future schedule slips. This feels like "flexibility" to the project manager, because it keeps her options open in the future.

The temptation to keep the buffer in the schedule must be resisted. Just as the overly aggressive deadline had to be resisted with transparency and honesty, the project plan must be kept honest in this case as well. The reason this is the truly flexible option is that the extra effort can then be reused by the organization, either to extend the current project or to apply it to a future one. Keeping that effort on the current project schedule denies it to the organization and limits its ability to develop other projects.

### DON'T CONFUSE "EASY TO DESCRIBE" WITH "EASY TO IMPLEMENT"

Many changes are very easy to describe in words. Yet most programmers can tell horror stories about being asked for a "tiny change" that turned out to require an enormous amount of effort. A manager asking for an easy-to-describe change will often assume that any project manager who balks at immediately implementing the change is being inflexible.

It is a common myth that having a software process—that is, deciding on how you are going to build the software before you actually build it—is inflexible. The feeling is that requiring that the software be planned and designed before it is built will prevent programmers from just jumping in and making any small changes that are needed by the organization. This is one of the most common complaints that project managers hear when trying to implement a reasonable planning process.

In fact, it is the planning process itself that provides the most opportunity for flexibility. It's very difficult to figure out which changes are easily contained, and which ones will have much larger consequences. Sometimes a change that seems tiny will require a large coding effort, while a change that seems large from a user perspective is actually relatively minor to implement but could have a lot of testing implications. The only way to get an accurate picture is by engaging the team and having them estimate the impact of the change—preferably using a change control process (see [Chapter 6](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch06.html)). Controlling the changes will give the organization the most flexibility.

# Manage Your Team

Many project managers—especially those who have been promoted from technical positions—feel like their primary job function is to understand the job each team member is doing. Often the best programmer, designer, or tester will be promoted into a management position. And in many cases, this is a very good choice, because it's important for a manager to understand the work being done by the team. Sometimes the project manager is also the direct manager of each team member, and in this case, he is especially vulnerable to certain management pitfalls.

Understanding the work that the team is doing is very important; however, the primary task of a manager is to have the right people do the correct work and, ultimately, to get the tasks done in the most efficient and least problematic way. The first instinct of a manager who got to where he is by being a good programmer will be to be the best programmer on the team. That's not what the team needs—they need someone to make decisions, provide guidance, and prioritize their tasks. If all he does is "get his hands dirty" by solving programming problems for his team, they will not only sense a lack of direction from their manager, but may also feel demotivated because their work is not valued.

In contrast, some managers understand that their job is to delegate. But while delegation is an important part of management, it must be done with a good understanding of the work being delegated. The manager may not be the best engineer in the group, or even be able to perform all of the engineering tasks he assigns. He should, however, understand the goals and limitations of each task, and be able to offer real guidance if team members get stuck or need help. That's much harder than delegating: while he's trusted a team member to accomplish the task, he must still understand enough about it to be useful if that team member encounters a problem.

Good managers usually feel a little guilty about being managers. They know that their people are good, and they want them to succeed. But this necessarily involves riding their coattails. As a manager, you might feel that in some ways you are not making a direct contribution by producing work products. This is a good feeling: embrace it. Recognize that your role is to "grease the wheels" by providing an environment in which work gets done. And the best way to provide that environment is to show the team that you trust them to do the work. Show the team that you are there for them when they need you. When you make decisions about the project, make sure that you are always fair, just, consistent, and predictable. That way, when people disagree with you, they can at least understand why you made that decision and will remain motivated and loyal to the project.

## Avoid Common Management Pitfalls

Poor managers are distinguished by their poor habits. They tend not to question their own authority, and they frequently don't have much faith in the people who work for them. They distance themselves from their projects, and tend to see their jobs as simple, intuitive, easy, and straightforward. A manager who acts this way projects hubris and arrogance; some people find that reassuring, but most engineers find it condescending. The best way to improve as an engineering project manager is to avoid these pitfalls.

The best way to avoid these pitfalls is to question each decision that you make:

* Is the decision based on facts, or are you automatically jumping to a decision based solely on intuition and gut instincts?
* Are you unfairly questioning your team's honesty, integrity, or skill?
* Are you making a decision that is politically motivated? Are you simply siding with the person you like better?
* Are you oversimplifying a task or avoiding its internal details out of laziness?

By understanding the root cause of many common pitfalls, a project manager can keep his team motivated and avoid bad decisions that lead to serious project problems. It requires constant vigilance to avoid those problems.

### DON'T MANAGE FROM YOUR GUT

There is a common belief among many managers that decisions should make intuitive sense. This is generally true. However, the converse—that all ideas that make intuitive sense are good decisions—is definitely not true. Software projects are complex. They involve many people serving in different roles, sometimes with multiple roles per person. They involve numerous tasks and subtasks, where each task may be assigned to several people. As a manager, you can't expect to intuit all of that complexity. Just because you, as the project manager, have the authority to make decisions about the project, that doesn't mean that it's your job to overrule people all the time. It's your job to understand the issues that face the team, and to help the team deal with those issues.

Think about it rationally: if a team member disagrees with a decision that you have made, and comes up with a well-researched and logical explanation for her disagreement, is it fair to dismiss her opinion simply because it does not immediately make intuitive sense to you? There are many things in the world (especially in complex engineered products) that simply are not intuitive to most people.

For example, it seems intuitive that doubling the staff on a project should allow them to complete it in half the time. However, in the real world, projects are much more complex: there is overhead in the extra communication, certain tasks on a critical path cannot be split, it takes time for new team members to ramp up, etc. But that doesn't stop many project managers from trying over and over again to throw additional team members at a late project, only to find that it becomes an even later project.[[\*](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch10s04.html#ftn.appliedprojectmgmt-CHP-10-FN1)]

Unfortunately, software project managers have to make decisions based on complicated information all the time. To make good decisions, you have to understand software engineering concepts and technological concepts that are not intuitive, and remain open to the idea that there have been recent innovations or changes in software engineering and technology that may contradict your current beliefs. This job is about being informed, not about feeling your way through problems.

A project manager must make many individual decisions: who to assign tasks to, how long they should be expected to take, whether to implement certain features or requirements, the dependency between tasks and software behavior, and many other design, development, and testing decisions. There is no way that even the best project manager can be on top of every detail in an average-sized software project. But these decisions still must be made. So how can you make them without simply relying on your intuition, but also without being overwhelmed by the details?

Luckily, your project team is staffed by competent software engineers who are capable of building the software. (If your team is not competent, you have bigger problems!) This means that you have at your disposal people who can help you make those decisions. You should enlist their help and work to understand the perspectives of all of the people involved in the project. When you make a decision, you must understand which team members it affects and take their perspectives into account. If you don't know those perspectives yet, ask the team members their opinions. Most people will be more than happy to help you decide the direction of their tasks, and you will almost certainly get better results because they participated in the decision-making process.

If you try to learn all of the details for every decision that must be made, you will find that your projects will quickly get bogged down, with everyone waiting for you to decide on at least one issue. But if you work with your team to make well-informed decisions, you can share that load...and everybody wins. That's why you have a team: so people can collaborate.

### DON'T SECOND-GUESS ESTIMATES

Many managers fall into a common trap when considering their team members' estimates: they automatically cut those estimates, no matter how reasonable or well researched they are. There are generally three reasons this is done.

One reason is that the organization already committed to an earlier date for the software, and changing that expectation is difficult or impossible for the manager. This means that the project was not planned properly. The solution is to apply the project planning and estimation tools and techniques to bring the project under control. If the estimate does not meet the needs of the organization, the manager has several options: the scope of the project can be scaled back; the software can be released in phases; the deadline can be renegotiated; resources can be added; or some combination of all of these can be done. The team will generally respect the decision, as long as it was clearly based on honest estimates and planning rather than an artificial date.

The second reason that an estimate may be second-guessed is that this second-guessing is a misguided attempt to motivate the team. For some reason—and nobody is really sure why some people believe this—there are managers who think that telling somebody that they can do a task in less time than they estimated will cause them to "step up to the plate." Somehow, knowing that their manager believes that they can do it is supposed to increase their productivity. Unfortunately, this sort of second-guessing becomes a self-fulfilling prophecy. As the team realizes that their estimates will always be cut, they will start padding those estimates. This can create an environment of distrust between the team and their manager.

The third reason managers will second-guess their teams' estimates is to force them to work overtime. By enforcing an overly aggressive deadline, some managers find that they can squeeze five, ten, or more extra hours per week out of all of their team members. This is especially dishonest, and it almost always breeds resentment. If the team is expected to work overtime—and in some cases, this is a valid and realistic expectation—that should be taken into account when the effort estimates are turned into a project schedule. This way, the team is not blindsided by the extra work and can plan their lives around it. (Sometimes managers forget that people have lives outside the organization!)

In all of these cases, the key to understanding how the team members react to second-guessing is to recognize that they believe their manager is sending them a clear message: he does not trust their estimates. The solution to this is to establish trust by making decisions in a transparent manner. (A good way to do this is to use a repeatable estimation process like Wideband Delphi—see [Chapter 3](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch03.html).)

This does not mean that the project manager does not need to understand estimates. It is important for a project manager to not only understand the reasons why the team estimated a certain effort for a task, but to question the estimate if it looks inaccurate, unrealistic, or seems to be based on incorrect assumptions. As long as the questions are reasonable and can be answered with facts, the team will generally respect them and work to answer them. If it turns out that the estimate is, in fact, unrealistic, the team will be glad that the project manager pointed out a potential problem with it!

### DON'T EXPECT CONSENSUS ALL OF THE TIME

Over the course of almost any project, there will be disagreements between team members. Some project managers make the mistake of expecting the team members to settle all of these disagreements, reaching a consensus on their own. The project manager treats all disagreements as if they were merely petty or politically motivated, rather than based on a genuine difference in opinion over some important issue that affects the project.

When two team members have a genuine disagreement, it is the manager's job to make a decision. That decision is going to leave at least one of the team members—and possibly both—unhappy. It is important to share the reasoning behind the decision with everyone, and to stand behind the decision. If it turns out to be wrong, it's the project manager's fault, not the fault of the person who originally proposed the solution or of the person who didn't fight hard enough for the alternative.

To make a good decision, the manager must understand both perspectives. It's not enough to just tell the two people to go decide among themselves: if they could do that, they would not have brought the disagreement up with their manager in the first place. Sometimes a compromise can be reached, but most team members are capable of recognizing when a compromise is available, and implementing it themselves.

If you treat each conflict as if it were a trivial or petty argument and tell your team members that it's their own responsibility to solve it, you are essentially asking one of them to acquiesce on something that he clearly thinks is important. That is unfair and divisive, and it makes both team members feel as if you do not care about their concerns or the project itself.

That's not to say that there are no problems that cannot be left to the team. Sometimes a problem really is petty ("Bill stole my stapler!") and the team members really should at least try to work it out between them before involving their manager. But even these problems can escalate, and if that happens, a concrete decision ("Buy another stapler") is the only way to make the problem go away. It's important for a project manager to learn to differentiate between trivial problems ("Someone keeps taking the last donut") and more serious ones ("Tom won't let go of his ridiculous database design").

Regardless of the magnitude of the problem, if two people on your team care enough about a problem to come to you with it, you should take it seriously. If you dismiss it and tell them that it's their problem to solve among themselves, you are making it clear to them that even though you are their manager, you do not care about the team members' problems (and, by extension, the project itself).

### AVOID MICROMANAGEMENT

When a manager is overly involved in each task to the point that she personally takes over the work of each person on her team, she is micromanaging. From her point of view, there are a lot of benefits to micromanaging her team:

* It endears her to the people at or above her level, because it seems like she always knows everything there is to know about what's going on with her team.
* She knows that no mistakes will ever make it out of her team.
* She does not have to trust the people who work for her to do a good job. Instead, she can review everything they produce to ensure that each work product meets her standards—and she will redo anything that does not meet those standards.
* It makes her feel very important, because nothing gets done without her.
* It allows her to feel superior to everyone who works for her.
* She gets to steal her team's thunder, taking credit for all of their accomplishments while blaming them for any failures.

Her micromanagement has a devastating impact on the people who work for her. They feel that they have no responsibility whatsoever for what they do. They are not trusted. If they produce poor work, she will fix it, usually without explaining what they did wrong and often without even telling them. They feel like they do not have any impact, positive or negative, on the final product. And they're right.

Many people will put up with this situation for a long time. They can continue to collect a paycheck. The job that they do is not particularly stressful, because any work that does not meet the organization's standards will be redone for them. They are not trusted to set priorities, make decisions, or do any aspect of their jobs. This is very inefficient for the organization, and very demotivating for the team members. While they will tolerate the situation, the team members are neither challenged nor fulfilled. Meanwhile, their manager is drowning under all of the work. Nobody is happy with this situation.

There are a few easy rules that will help you avoid micromanagement:

Don't expect to review everything

Many people think that to be an effective manager, you have to have read, reviewed, and redone all of the work of the people who work for you. A good manager will spot-check the team's output, but reviewing (and possibly redoing) every piece of work that the team creates is a terrible use of a manager's time. Delegation requires trust; if you do not trust your team to do their jobs, then you should fire them and replace them with people who you do trust (or don't replace them, so the organization does not have to pay their salaries).

Don't fall into the "hands-on manager" trap

There is a general perception in the technology world that management is not an actual job. It's often believed that competent engineers manage themselves, while their incompetent "pointy-haired" bosses just get in the way. This is simply untrue. Competent engineers can be trusted to produce good requirements, designs, test plans, and code; their focus is not on prioritizing or managing teams of people. They can't do your job for you, so don't try to do theirs for them.

Many managers assume that because they are responsible for the work that their team produces, they should be able to do all of it. That's just not true—the individual team members have time to build up their expertise in developing the software, while you only have time to manage the project. Instead of trying to fill in as a technical team member, work on building up your project management skills.

Use transparency to your advantage

Some people fall into the trap of thinking that the job of project manager consists of constantly bugging each team member for status reports. Team members have trouble with this. They are surprised and unhappy that their project manager doesn't know enough about the project to glean even the most basic status; they also feel that they are not responsible for reporting their status upward. The project manager always goes to the team members, so they don't ever feel the need to report problems unless directly asked.

If all project plans and work products are kept as a matter of public record, the team members don't need to deal with a project manager constantly bugging them for status reports. If the project has transparency, each team member is responsible for his own status communication.

If your team is falling behind, don't just ask them for their status—this will encourage them to give you excuses. Instead, gather the status yourself using the documents that you have made public, and ask them about specific problems. Transparency only works if you make it clear that there are consequences for poor performance, and that poor performance is evident from public documents. Given this, people will see what's required of them in order to do a good job.

Don't be afraid to let your team make mistakes

People learn from their mistakes, and it's important that they be given the opportunity to make them and take responsibility for them. If they are going to grow professionally, it is going to be through their own experiences and through trial and error. Without this, team members will never see that their success takes effort to achieve. If a micromanager simply corrects all of their mistakes and redoes their work, they have no incentive to learn or improve.

It's okay for your team to make mistakes, as long as you're on top of it. The way to stay on top of mistakes is through the peer review tools. This allows team members to share information and help each other to grow to a standard that is in line with what the project needs and what the organization requires.

### MAKE YOUR MISTAKES PUBLIC

If you make mistakes, you need to communicate them to everyone around you. It's okay to make the wrong call. Good managers recognize when they have made mistakes and correct them. The best managers publicize those mistakes so that everyone can learn from them, and so no one is blindsided by them.

When a team member makes mistakes, the project manager should share in the responsibility. Many project managers don't realize that they are culpable for the mistakes made by their team members. The reason the manager is culpable is because he assigned the task to the team member, allocated the resources to do it, set the priorities for the project, and set and communicated the deadlines—all of which were contributing factors.

For example, if somebody makes a bad decision because she failed to understand the project priorities, the project manager shares the responsibility for the error. That doesn't mean the project manager is solely to blame—if there were 20 people in the meeting where that priority was communicated, and 19 of those people understood it, the one person who ended up making the mistake should have spoken up at the time. But it still means that the project manager was not entirely clear, and failed to communicate the project priorities to everyone on the team. So when that mistake gets made, it's still partially the project manager's fault.

Just as it is okay for team members to make mistakes as long as they learn from them and corrective action is taken, it is okay for project managers to make mistakes as well. That's not to say that there are no consequences for mistakes—a serious mistake can lead to delays, which could lead to reprimands and even failed projects. As a project manager, if you find that one of your team members has made a mistake, it's your job to figure out what role you played in that mistake. And when you communicate that mistake up to senior management, you should highlight your role in it and try to shield the individual team members as much as possible.

This is very difficult to do. It's human nature to blame others. But by taking the blame yourself, you protect the team and keep them motivated, and help prepare them to recover from the mistake. If you stick your neck out for the team, they will know it, and they'll be much more loyal to you. On the other hand, if you let the blame roll downhill, the team will resent you and begin to work against you.

Some project managers don't think this is fair. They feel that if someone makes a mistake, it's that person's responsibility to take the blame for it. But the project manager is in a position of authority, and just as other people are accountable for their individual responsibilities, the project manager is accountable for everything he is responsible for. And he's responsible for the entire project.

### AVOID LISTS

Some managers do little more than hand their team members lists of tasks or action items. When a team member is handed a list of tasks, but has no understanding of the rationale behind each action, he does what he can to complete the task. But without a real understanding of the needs that drive the project and the rationale behind each task, he will often make basic mistakes that would be obvious if he were given the proper context. There will always be decisions that a manager cannot predict and put on a list; without context, a team member has little chance of making those decisions correctly.

It's easy for a team to feel comfortable working from a list. It means that they are not responsible for anything other than this list of tasks. They don't have to think about overall project goals or the bigger picture. Most importantly, they don't have to make decisions. Accomplishing everything on a list of tasks is gratifying—a team member can go home at night knowing his job is 100% complete. But someone who feels responsible for the project, and not just his own tasks, knows his job is not really complete until the software is delivered and accepted. Sadly, once a team member understands the big picture, he feels like he's never done.

Your job as a manager is to get everybody on the team to see the big picture. The vision and scope document is a valuable tool for this, as are the rationale sections of the use cases and requirements. These tools allow the team members to more fully understand the context that surrounds the work that they are doing.

Software project teams are made up of smart people. It's far better to leverage their minds than to treat them like robots. Many people like to throw around the term "grunt programmer," as if there were a lot of programming tasks that were little more than cutting and pasting program recipes. But even the lowest impact programming tasks involve decision-making on the part of the programmer.

## Accept Criticism

There are two ways that managers encounter criticism from a team member. One way is when the team member disagrees with the way a manager wants work done. The other is when the manager disagrees with how the team member is doing the work. Dealing with criticism is a potentially demotivating situation, but it's also a potentially encouraging one if handled well.

Sometimes the team members solve a problem differently than you would. As a manager it is important to recognize the work that has gone into these solutions, even if they contradict your preconceived ideas about that work. Being able to accept the team member's criticism of your solution means that you are making decisions in the project's and organization's best interests, and motivates your team to keep thinking their way through such problems in the future.

Everybody solves problems differently, and it's a fact of life in software engineering that most problems have many correct solutions. When you ask a team member to solve a problem, it's likely that she will come up with a correct solution that is different than the way you would solve the problem. If a member of your team offers a solution that is technically correct, but you don't accept it because you would do the work in a different (but equally valid, or even slightly more efficient) way, the team member will feel crushed.

A good manager's default action should be to accept, not reject, ideas. You must take it very seriously when you reject somebody's work, and when you do, you should always explain and defend your decision. There are many good reasons to reject a team member's solution. Sometimes it's incorrect, and sometimes it's not well thought out. But people will become very attached to such solutions, even when they are dead-on wrong. In those cases, you must be willing to stick to your guns. When you do, it must be for the right reasons, and you must take the time to explain your reasoning.

Criticism goes both ways. Sometimes a manager will want the team to do their work one way, and some team members will disagree. In fact, sometimes the entire team will disagree with a decision and come to the manager en masse. In this case, it is very tempting to just roll over and give in; it is equally tempting to refuse to even consider their opinions. Neither of these options is good, because no real discussion takes place in either case. Instead, a good manager will come up with a real justification for why he wants the work done that way, and will expect the team to do the same. If there is a real, verifiable reason for going with one alternative, everyone should be able to see it. And most importantly, the manager should show that he considered the argument, even if he essentially rejects it.

Ultimately, you won't be able to make everyone happy. It's always better if everyone can agree, but there are many times when there is a genuine difference of opinion. In this case, the manager is within his rights to pull rank. However, if he just rejects an argument outright or ignores a valid argument just to get his way, he is abusing his power, and his team will resent him and try to figure out ways to work around him. They will also avoid coming to him in the future, opting to apologize later rather than ask permission now.

Another way to help team members accept your decisions is to have written guidelines that they can follow. If you can point to a published document that guides the way that your team does their work, your team members will recognize that and respond to your consistency. It's much easier to work with a manager who is consistent and predictable than with one who may randomly reject ideas with no real justification. The tools in this book are examples of the kinds of guidelines that a team can adopt. For example, a manager may have a written guideline that says that every programmer should follow the Subversion basic work cycle (see [Chapter 7](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch07.html)). Then, even if a programmer feels that it's not her responsibility to merge changes that occurred since a file was checked out, the manager can refer back to the guideline and show that he is being consistent in his decision to have her merge her changes.

## Understand What Motivates Your Team Members

Talk to your team about their goals. If an employee's goals are incompatible with his company's goals, he should not be working for that company. However, each person's goals go beyond simply finishing the current project: people want to move ahead in their careers, and part of your job as project manager is to help them achieve their professional development goals. The organization gains when employees improve, because a more experienced employee will deliver superior work. A team with people who have more experience can take on more complex projects and come up with more creative solutions.

People work for money. For some reason, many bosses feel uncomfortable with this—they pretend that people work out of loyalty, love of the job, or blind devotion to the organization's goals. This leads to an unfortunately pervasive attitude where managers act like their employees are lucky to have jobs. Compensation also comes in many forms: in addition to money, some organizations will give flexible hours, training, books, stock options, free lunches, senior titles, access to new technology, or other perks in place of money. But in all of these cases, people need to feel that they are being fairly compensated for the effort that they are putting in.

Another motivator is loyalty. Many people naturally develop some loyalty to the organizations where they work—it is human nature. This is why teams of people who are poorly managed and undercompensated will still work 80-hour weeks on projects that are clearly doomed to failure. Unfortunately, it's very easy to redirect loyalty, especially through dishonesty. In some cases, a poor manager can keep secrets from the team and lie to them about organizational decisions, in order to redirect the team's loyalty from the organization to him. In other cases, senior management themselves can, through lying, incompetence, and obvious lack of appreciation, lose the team's loyalty.

## Don't Be Thrown by Dishonesty

People lie. They will say that they have completed things that they haven't, or that they understand things that they don't. They will make commitments, and then claim they never made them. Having a dishonest person working on a project is possibly the most difficult situation a project manager can face.

There are some things a project manager can do to discourage dishonesty. By keeping all work out in the open and admitting your own mistakes, you can create an environment where people are more honest. But this only goes so far—sometimes people lie, and you'll have to deal with it.

The best-case scenario is one in which you have evidence that directly contradicts the lie. If you find that somebody is lying, you need to present him with that evidence. The purpose is not to make him feel bad; rather, it is to help him understand that it's wrong, and that he shouldn't do it again. Don't get caught up trying to understand why someone is being dishonest—it could be a misunderstanding, it could be malicious, or it could be something else entirely. Sometimes the person doesn't even realize that he's lying. The key is to have enough information available so that you can set the situation right and keep it from threatening the project.

Unfortunately, in some cases, there is no evidence to counter the lie. When this occurs, there may be nothing that you can do about the situation. If you think that someone is lying and you don't have evidence, you can set about collecting that evidence. Usually a lie is about a commitment that was made: the person may have agreed to do a certain task in a certain way, and is now claiming that she never made that commitment. Information about the commitment may be in an email, a project document, or a task definition in the project plan. But if the commitment was less formal (such as a verbal agreement), there may simply be no record of it.

If there is not enough evidence, you may have to let the lie pass and live with the consequences. This is a very frustrating situation. In this case, your job is to improve the way you manage your commitments and those of your team, in order to prevent problems like this from happening in the future. You can collect better information, change your expectations, help people feel more comfortable letting you know if there are problems, and, in extreme situations, avoid working with people who have trouble being honest.

### NOTE

More information on commitment management can be found in Managing the Software Process by Watts Humphrey (Addison Wesley, 1989).

## Address Performance Problems Early

It is difficult to effectively manage teams without defining their goals up front. The best way to do that is to involve each person in the definition of his or her own goals. Each of these goals should be specific, measurable, should pertain to their work, and should be attainable. One effective way to do this is to work with each team member to develop a performance plan, which is simply a list of goals that the manager and team member agree to.

People need to feel that they understand what is expected of them. The purpose of the performance plan is to set standards that are fair and attainable, and that are developed with the involvement of the team member (when possible). Your team members will feel more comfortable with their jobs if they feel they are being asked to meet reasonable goals and perform within their abilities. On the other hand, when someone does not know what is expected of him, you may feel he is doing a poor job when, in reality, he simply does not know what you expect of him. (You may not know, either—which is another reason a performance plan is useful!)

The manager should measure each team member's progress in meeting the goals listed in the performance plan. If the organization's operating environment changes, the manager should work with the team members to change those goals.

In many organizations, team members do not report directly to project managers; rather, they report to people who manage development or QA groups in the organization. However, a project team member whose goals are poorly defined or in conflict with the objectives of the project can threaten the project's success. When the success of the project is threatened, it is the project manager's responsibility to remove the threat. This may require that the project manager help the direct manager establish a performance plan.

You may find that your project team members' professional goals, set by their direct managers, conflict with the objectives of your project. For example, a programmer on your team may feel that meeting the deadlines for delivering code is more important than carrying out code reviews and unit testing, which he sees as optional, "extraneous" activities. By failing to do code reviews and build unit tests, he meets his personal deadlines, but he causes the project to be late because it spends more time in testing. If the programmer reports to a development manager, for example, it is your job to bring this up with that manager. One way that you can suggest that he fix the problem is by building a performance plan that includes goals that are quality related.

It is important to correct performance problems as early as possible. Many project managers make the mistake of waiting until the end of a project to try to address performance issues. If a team member is not doing his job properly, the project manager may not have the authority to fire or discipline the team member. But he can have that team member removed from his project, if he is unable to correct that person's behavior by either dealing directly with the team member or going to the direct manager. By addressing the problem as early as possible, the project manager limits the risk to the project.