**Chapter 11. Managing an Outsourced Project**

**Managing projects is hard**. Scope creeps, changes go uncontrolled, defects are introduced, schedules are delayed...and that's all in your own organization, where your software engineering team is right down the hall. Imagine how difficult it is to get even these results when your team is in another organization in an entirely different building—and possibly in a city halfway around the world! When you hire a company outside your organization to build your software, you open up yourself, your project, and your organization to exactly these problems.

Unfortunately, that straightforward reasoning seems to be lost on many people. The fact is that outsourcing is risky, and many people find that their projects go awry. Gartner, a respected research and consulting group, recently (at the time of this writing) published a report that predicted that half of IT outsourcing projects in the next 2 years will fail, and that 60 percent of organizations that outsource customer-facing processes will find that hidden costs and customer problems have wiped out any cost savings. This implies that leading an outsourced project requires a different set of skills than most project managers are familiar with. If you are used to working with an in-house team, you personally will need to change your approach to project management if you want to get your outsourced project done right.

There are a lot of overly optimistic books, articles, and papers written about outsourcing. Mary Lacity and Rudy Hirscheim propose reasons for this phenomenon in their book *Beyond the Information Systems Outsourcing Bandwagon*. They point out that much of the outsourcing literature is written during the "honeymoon" period, after the contract is written but before any project milestones are met (or blown). Many only report projected savings, not actual savings. And most of all, they point out that only the successes tend to be documented, because few organizations want to publicize their mistakes.

The truth is that the project manager for an outsourced project faces all of the challenges she would face on a project developed within the organization, plus a slew of additional difficulties. However, if she is able to navigate these issues, she can lead a successful project.

**Prevent Major Sources of Project Failure**

By getting involved in your software projects and not leaving all of the decisions up to the vendor, you can prevent many of the most common causes of outsourced project failure.

**Get Involved**

There's a broad misconception that it's possible to somehow extract the work from an outsourced project team without getting involved in the day-to-day management of the individual team members and their tasks. The conventional wisdom goes something like this: you're paying the vendor to handle all of the management overhead, so they should also be able to handle any personnel management problems that arise. This might work in some instances, but in other cases, it leads to a client who is unhappy with the vendor, and to a vendor who does not really interact with the client.

In fact, that very management overhead at the vendor can be a major source of problems on your project. For example, an in-house development team in an organization can standardize on a single language and platform; a vendor, on the other hand, must be ready to take on a broad range of technologies for many different clients, most of which will not have any application whatsoever to your project. To accomplish this, the vendor has an incentive to keep all of the programmers on the team cross-trained in multiple technologies. One way to do this is to make sure that each programmer is only assigned to any one project for a short period of time, in order to expose him to many technologies. This means that long-term engagements can be difficult to set up: many of the people on your project will see that by staying on your team for a long time, they are falling behind on the technologies that allow them to advance within the vendor's organization. This does not necessarily work against you, but it certainly does not work to your advantage.

Similarly, most good outsourcing vendors will put effort into cross-training other people in their organization on best practices learned from each client. While this can be good (because it can potentially help them address your needs better), it also means that some of your project team members will be allocated in part to helping other project teams internally at the vendor.

In other words, there are many ways that your needs and the needs of the vendor are not perfectly aligned. This makes sense—you have two different organizations, with different businesses and goals. But this difference introduces distance between the client and the vendor, which can lead to some specific ways that many outsourced projects fail.

The most common response to this—and the most serious mistake that a project manager can make when working with an outsourced project—is to assume that it's the vendor's responsibility to fix every problem that comes up in a way that will guarantee that the software gets built properly. There is very seductive logic here: "I'm paying the bills, and the vendor will lose my business if they don't get this right, so they *have* to take care of everything!" This attitude fails every time. The vendor does not have enough information to build the software properly, and a hands-off project manager who leaves everything to the vendor will find that the software that is delivered does not meet the needs of his organization.

**Constantly Communicate Project Goals**

The people working on your project have different goals than you do. Many of them may see the tasks that you give them as simply an opportunity to expand their knowledge. Others are completely devoted to the work that you have given them, but they don't necessarily see the context in which you have asked for it; all that they know about the project is what you have told them, and they don't (or can't) have an intuitive understanding of your organization's needs because they don't work there.

What this amounts to is the fact that you are the only person working with the project team who has your organization's needs and goals in mind. As a result, outsourcing teams are much more susceptible to the "Do what I mean, not what I say!" problem. An in-house project team almost always has a grasp on what it is that the organization does; people get an enormous amount of context just from working at an organization. Team members at an outsourcing vendor lack this context, and, unless you work to provide that context for them, that can lead to serious project problems.

For example, programmers who have worked at an accounting company for any length of time will have naturally absorbed some knowledge of accounting. When these programmers are given a task, they will immediately see why the organization needs that task done in terms of that accounting knowledge. They will be much better equipped to fill in the gaps when presented with incomplete requirements.

If the same task is given to a team at an outsourcing vendor, the programmers will not have the same background or knowledge. Even if that vendor has experience with other accounting projects, the team members do not spend each day working at an accounting company, talking to other people who do accounting, or reading the company newsletter about the latest clients. They do not come to the table with the same expectations about the software that an in-house team would.

This is why it is especially important to communicate the goals of your project to your team all the time. You need to personally make sure that the team members understand your organization's needs, and that the tasks they are performing are in line with its goals. Since they are not immersed in your organization's culture, you have to be the ambassador of that culture, so that they are always kept on track. You have to act as a rudder, constantly steering the team toward the goals of the project. That could mean that you need to have daily discussions with someone from the team. You may need to spot-check work from selected team members to make sure you are getting what you think you are asking for.

This can be very difficult and time-consuming, but it is easily the most important thing that you can do to make sure your project does not fail.

Many outsourced projects fail because their project managers fail to understand or do anything to compensate for this situation. They often blame the vendor for not understanding what they are saying. This is frustrating for everyone involved: the project manager feels betrayed by the vendor, while the vendor feels like the client did not adequately communicate his needs or goals. Nobody is happy with a failed project.

But despite how many project managers feel about their projects, in most cases this is not the fault of the vendor. Rather, it's a fact of life due to the way the outsourcing industry is structured. Each vendor has its own business to run, and it's absolutely understandable and expected that they would want to train people and encourage them to grow within their organization. It often falls to the project manager of the outsourced project—in concert with the management team at the vendor—to balance the vendor's needs with the project's goals. This means that you must form a relationship with their management that allows you to deal with that problem and to reach an acceptable compromise.

The more you are able to integrate the outsourced team with your organization, the more context they will have. If you are able to dedicate multiple people at your organization to communicating with the outsourced team, the team stands a better chance of understanding the complexities of your organization's environment, and of ultimately meeting your organization's needs. It doesn't always fall to the project manager alone to communicate the culture, context, and needs of the client organization. The more people you can involve in that process, the better it will be for the outsourced team. If your organization is willing to put in the effort, they can build a sense of teamwork between your organization and the vendor's that would not be possible to build on your own.

**Make Sure the Project Is Estimated Well**

The further away you get from a task, the easier it seems; the devil is usually in the details. Outsourcing allows you a lot of distance from your projects. For example, if your own team is making estimates , you often expect those estimates to be examined and questioned by senior management. But an outsourcing company does not have the same checks, and also does not have the same implicit trust. In some ways, you're far more likely to distrust them; but you're also much more likely to have little or no visibility into the way they run your project.

Sometimes it's the client's fault that realistic goals are not set for the project. There may be a lack of due diligence on the part of the client contracting the outsourced services—for example, choosing companies based on cost only. Some companies may be cheaper because they don't understand the project being proposed, while some may be cheaper because they just aren't very good. (For some reason, clients don't care what the reason is for a very low price until they see the final product, or lack thereof).

On the other hand, sometimes it's the vendor's fault. Vendors tend to promise things they can't deliver. (This shouldn't be *too* surprising—most software engineers have experienced projects where the promised deadline was unrealistic.) Many vendors are perfectly aware of the myths about outsourcing, and are happy to let you continue to believe them. ("Your project can't fail because there are many people sitting in the wings, just waiting to jump on if the project starts going downhill."[[\*](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch11.html#ftn.appliedprojectmgmt-CHP-11-FN1)]) When you're talking about a software project, it's going to be a long time between when the contract is put in place and the point when you figure out that the project is not progressing—especially if you don't have good checkpoints in place.

One effective way to prevent the vendor from taking on work that the team cannot perform is to understand their capacity from the outset. Ask them to show you the results of a project of similar size. (If they have never taken on a project of similar size, you may want to switch vendors!) Get involved in the estimation process, and make sure that the people who are going to do the work buy into the estimates that the project plan is based on.

Don't be afraid to meet with the vendor's project team and hold your own estimation sessions, once the project team is assembled. It is not uncommon for a vendor to have a separate estimation team that provides estimates when a contract is being negotiated; this team may provide estimates that are sufficient for a contract, but insufficient for planning your own organization's goals. By getting personally involved in the estimation process, you can ensure that your project plan is better grounded in reality.

# Management Issues in Outsourced Projects

There are some important ways project management is different for an outsourced project than it is for a project developed in-house. It's not just the relationship with the vendor that's different; it's also the relationship between you, your management at your own company, and your team members. By paying special attention to transparency, information sharing, and communication, you can make sure these relationships lead to a successful project.

## Actively Manage Your Project

If you have a relatively small project with well-defined requirements, known acceptance criteria, and a specific deadline—in other words, a fixed amount of work in a fixed period of time—then the project will probably work out fine. The success of a project like this hinges primarily on the technical competence of the programmers and on their ability to work together as a team. This is especially common when an organization that has never developed software before needs a specific piece of software written and does not want to build up its own IT infrastructure.

If you are a project manager at an organization that makes software as part of its core operations, then you are probably not in this situation, and your outsourced projects will probably not be this simple. They are much more likely to involve an open-ended commitment (one that, in some cases, could go on for years). Many organizations that outsource their work do not have a good grasp on their requirements, and often have not put much thought into what they need from the project or the vendor (other than "working software"—which, in practice, is not an easy thing to pin down).

The truth is, many outsourced projects don't go well. What's more, the clients at those projects don't necessarily realize that their projects have started to go bad until they receive a piece of software that does not do what they need it to do. This situation is harder to prevent here than it would be in an in-house project, for several reasons. If you are getting a much cheaper labor cost than you would for your own employees, then you might have a much larger project team than you are used to managing. You might also be used to having a lot of visibility into how your projects are going, because your in-house programmers talk to you routinely and give you (and your users and stakeholders) a lot of status updates.

When you are managing an outsourced project, the status doesn't readily present itself. If you have a good management structure in place, you can trust your delegated managers at the vendor to relay the status to you. However, the most reliable way for you to get a good handle on the status of your project is to collect it yourself. If, for example, you are building a project, you should ask for nightly build reports and unit test results. You should track the lines of code produced on a daily or weekly basis, and you should have access to a defect tracking system with metrics. Numbers like that can give you regular snapshots of the health of the project.

You should know the names of the people on your team. Ask for a CV or resume for each person. Have regular conversations with at least one representative for every project—minimally, you should talk to that person weekly. But if you really want a handle on how your project is going, you, or someone in your team, should talk to someone from the outsourced team every day—the same as you would if they were in your office! (Luckily, with instant messaging, it is very easy to keep in touch with a large team by adding them all to a buddy list and spending time each day shooting messages to them.)

One effective way to make sure that you will get all of the information that you need to run the project is to set up a communications plan with the team lead at the vendor. Make it clear from the outset of the project exactly what information you need them to gather for you. If you do this, it is very important that you only ask them to gather information that will actually be useful to you, and that you use it and review it with them: if you do not use the information that they gather (or if they do not see how it will be useful), then it will just seem like busywork to the team, which is demoralizing and counterproductive.

If you don't make sure that the people at the vendor know why you are asking for this information and what you are using it for, it's very easy to build up an environment of distrust with your outsourcing vendor. The only way to combat that is with transparency . In other words, somebody at the vendor is going to spend half a day each week gathering data for you. Make sure that you respect that time.

The best possible scenario is when you've set up standards that let the people at the vendor monitor themselves. You want the team to be as autonomous as possible, while still being productive and giving you information that you need to monitor the project. Your goal should be to have the team assist you in managing the project without losing control of it. One way that you can do this is by setting up an inspection process where the team can inspect documents and report the results to you (see below). If knowledge that you need to know has been properly transferred, you can write a quiz, and have all of the people on the team take it independently and send you their results. The key is that you must know at all times what the team is doing, and that it's in line with what you want them to do.

## Share Information with Your Management

Many senior managers think that software development should be free. They think their internal IS departments are overpaid. Now they're cutting a big check to an outsourcing company, and they expect everything to be smooth and easy; there should be no problems or difficulties whatsoever. If you're the project manager on a project in this environment, you are set up for a thankless job.

When senior managers have unrealistic ideas about outsourcing, a project manager's successes will almost certainly go unrewarded because there is already an expectation that outsourcing is easy. If you do well as the project manager, the credit will go to the outsource company. What's more, in an environment like this, there is little incentive not to fail—if you do, it will be blamed on the outsourcing company. People are already biased against outside organizations handling their business; nobody wants you to waste money, but they won't blame you for somehow being "snowed" into thinking that another company could do as well as your own organization.

On the other hand, many senior managers have a much more realistic view of outsourced projects. They realize that they are difficult to manage, and that they require a lot of work and overhead. In this case, it is even more important that they are kept in the loop; you will need them to support you in case you make any controversial decisions, if you need further funding, or when you need their approval.

Regardless of the attitude of your senior management toward the project, it's always a good idea to keep them informed of everything you are doing. This means that you need to constantly go back to your own senior management and make sure that you still have their buy-in. If your boss thinks that managing an outsourced project is easy, you will have a very difficult time explaining why you spend so much time managing it. You must make sure your organization's managers understand what it is that you and your outsourced team are accomplishing, and how you are dealing with them on a day-to-day basis. You are the bridge between your organization and the vendor; it is your job to bring transparency to the process. Just as you have to focus on constantly steering the vendor team into meeting your organization's goal, you also need to constantly steer your organization's management so that you are always apprised of their goals and they have visibility into how the vendor is meeting those goals.

To provide adequate transparency into the project, you must give status reports to your organization's senior management. Any metrics that you use to track a software project will be useful; you should make them available. Encourage your management to visit the outsourced team and meet with the vendor's management. All of the methods in [Chapter 10](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch10.html) for making a project transparent should be applied. In this way, you can make sure that your management can actively get information whenever they need it—and you can help them understand just how difficult the job of managing an outsourced project is.

A large part of establishing an understanding with your managers is sharing the issues that you are resolving, so they understand the effort you are putting into your project—especially when it comes to managing the team. It may seem obvious to your senior managers that the team doesn't work for you, but it's not obvious that this can create its own set of problems. There's a difference between being someone's client and being someone's boss. You don't set the performance goals of the software engineers; it is rare that you even know what they accomplish in their careers.

## Build a Relationship with the Vendor's Management

You must have a good relationship with the entire upper management of your vendor organization. You must know who to escalate to if things go wrong, and you need to be able to trust them—and have them trust you as a credible and knowledgeable source of information for your project. You have to partner with the management of your project at the vendor company. They need to understand your goals. And most importantly, they need to understand that when you ask them to change the way they do their work, you are doing it to help them continue making money from your organization.

It is very important that you maintain this positive, cooperative relationship with your vendor's management. This is not always easy. There are times when the vendor has procedures in place that make it more difficult for you to see what's going on in your project, communicate with your team, or do your work in other ways. When this happens, don't just go in and make a bunch of changes. Instead, carefully discuss the problems and reevaluate the procedures that caused them. There is nothing wrong with the vendor having a business to run. Vendors need to keep their employees happy and challenged. Sometimes there are situations in which the vendor's goals are simply different than yours, and you must reach a compromise by being open, honest, and transparent.

One common mistake that many project managers make is to set up a complex or convoluted escalation process. Some vendors come to the table with these escalation procedures prebuilt, inserting a layer that blocks communication between you and your team. A policy that puts an escalation process in place will typically require that a team member first talk to the project lead and then to the manager at the vendor (and often one or two other people) before they are allowed to communicate directly with you. (This time-consuming process is often put in place specifically to bolster the management hierarchy at the vendor.) Often, the escalation procedure resembles the playground game of "telephone," where a message is passed from person to person until it is essentially unrecognizable. This is not a good way to communicate, especially when a problem is serious enough to warrant the involvement of the vendor's senior management (who will not know the specifics of the problem and are therefore more likely to obfuscate it).

Some people are more comfortable with an escalation process in place than without it. Middle managers at a vendor like it, because they can intercept problems that might be potentially embarrassing (such as an incompetent team member). They can better manage the client's perception by only allowing the "good" questions to be asked: for example, they can tweak the questions to make them seem less negative. Many project managers also like the escalation process, because it provides distance (and cover) if the project starts to go wrong. The further they are from the project team, the less culpable they are for its failures—it gives them a sort of "plausible deniability."

There are often good reasons for this. Many outsourcing clients are very hands-off and do not want to be "bothered" with the day-to-day operations at the vendor. Sometimes that makes sense, like when the client has little IT or project management experience. Many clients who have outsourced work honestly can't handle the idea that any mistakes have been made, and, even though that's a completely irrational and unrealistic expectation, they will use it as a reason to start renegotiating or dismantling contracts. These procedures are there to protect the vendor from crazy or irrational clients—and there are an awful lot of those in the outsourcing world!

However, since you are a good project manager and a reasonable person, you can work with the vendor to adjust things so that they work better for everyone involved. If there is a convoluted escalation procedure in place at the vendor, you are perfectly within your rights as a client to modify it or, even better, dismantle it entirely. This is okay, even if that's the "regular way" that clients interact with your particular vendor. The benefit of doing things the "regular way" does not outweigh the increased chance of miscommunication, scope problems, and requirements problems. Provided you take responsibility for your part of the end deliverable, you should be able to work with the team in the way you are most comfortable.

In addition to escalation, there are other procedures at the vendor that might not suit your method of working. Their system for performance reviews might reward your team members for things that are important to the vendor's organization but are not necessarily important to your project. For example, they might be rewarded for years of service at the vendor. It looks great from the vendor's perspective to have a team of "seasoned professionals" (who may have been around for a long time, but may not necessarily be stellar performers). If a vendor can quickly assemble a team of people with many years of experience, they can use this as a sales tool to get more lucrative contracts. Therefore, when they are recruiting potential employees, some vendors might offer a compensation package that rewards years of experience over knowledge. What's more, some vendors work very hard to prevent attrition because it looks worse for clients, and so will work very hard to keep from losing employees who have seniority—even if these employees are not all that great at their jobs.

It's not just hiring and firing practices that might not suit your project. There may also be trouble with the specific ways that vendors reward employees. It is common for vendors to reward employees for seniority with things like increased responsibility and access to new technology. This can be bad for your project: it is very unlikely that you will need to change your team members' responsibilities or switch technologies over the course of your project, which means that you are cutting your team off from what they traditionally expect as a reward for seniority. In other words, if you are developing a piece of software in Visual Basic, the chances are that you will not change to Java over the course of your project. If people on your team feel that they are falling behind in their technical skills, they will look to switch to another project at the vendor.

Look around your own organization: you can probably think of one or two people who have been there a long time, yet are not great employees! There's no reason to think that this isn't also true at the vendor. The bottom line is that a team member with years of experience in IT or seniority at a vendor is not necessarily competent. This means that the less you rely on the vendor's management to assign responsibilities to and reward your team members, the more control you have over your project. The best way to handle this is to set up your own system of rewards. For example, you can offer a bonus based on actual performance rather than seniority. To do this, you need to have a good handle on how well each person does her job—which, once again, means that you need to communicate with them. This works especially well if you personally assign tasks to the team members, rather than relying on the vendor's management to do that for you. This will require that you do your own project planning (see [Chapter 2](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch02.html)), and that you find ways of objectively measuring the performance of the people on your team.

It's not just processes at the vendor that need to be adjusted for your project. You may need to adjust them in your own organization as well. For example, your organization may have security policies or intellectual property policies that make it more difficult for you to manage your projects. These are problems you would normally never have to think about, but, since you are working with an outside vendor, you may now have to take them into account. Sometimes companies do not have policies in place to share information between the client company and a subcontractor. This slows down the process of creating software and sometimes makes it impossible for the expectations to be met.

Many project managers find it difficult to communicate project priorities to the team. Just as you might be the sole point of contact for your organization, you may be working with a single point of contact in the vendor's company who handles your requests. They may misinterpret you, or never actually understand your goals. Unless you negotiate for it, you may never actually have access to the team.

## Build a Relationship with Your Team

You don't have the same kind of relationship with the team that you would with a team in your own organization. As project manager, you do not directly hire the resources assigned to the project. You can veto any team member, but you do not have immediate access to employment history, performance reviews, personnel files, salary information, or other important information you would usually need to make those decisions. If someone does a bad job, your only recourse is to have him moved off of your project. It's likely that he is never even told that there was a problem; he's probably just been told that he's been reassigned. This means that your team could contain people who were removed from other projects in the past due to performance problems and don't know it.

You have to gain the respect of the people you have hired to do your project. The simple fact that you're paying the bill doesn't establish you as the ultimate authority in how the vendor should manage the project. It's your job to establish yourself as a credible partner and to work with the people at the vendor, in order to make sure they meet your goals (and to assure them that you respect theirs!).

It's a common misconception that, because you are paying a contractor, you have more control over the work product. Somehow people assume that by simply writing a big enough check, a vendor will snap to attention and build exactly the software they need. After all, their payment depends on the client's satisfaction, right? So the vendor must be doing everything they need to do in order to satisfy the client. This is an odd attitude: you also pay your employees, and their performance reviews are dependent on your satisfaction as well, but that does not guarantee that all of their projects will succeed. The truth is that outsourcing projects are at least as likely to have problems as in-house projects. And when they do have problems, you have only a few possible recourses. This means that you actually have less control over the performance of individual outsourced team members and their work products than you would with an in-house team.

Being the sole point of contact with the subcontract team is a big change for many project managers. When you are the project manager for an in-house project, your team has access to the entire organization: they can talk to managers, stakeholders, users, etc. This is not the case on outsourced projects, where the vendor must route all communication through the project manager. This introduces many potential pitfalls into your project. The most costly problems involve the project scope and software requirements. If you do not describe the project adequately, you will end up with software that does not meet the needs of your organization. An in-house project team has many more opportunities to catch these problems.

This requires that you pay a lot of attention to your project team members, possibly more than you would to an in-house team. This is paradoxical, as most project managers are more likely to spend time with an in-house team than they would with a team at a vendor. It's not only your job to communicate your company's needs to the team, it's also your job to understand the needs of each team member. For example, if your team member needs clarification—and it is essentially impossible to run a project where a team member does not need clarification!—you are that person's only resource for gathering the information. If you do not provide a communications path from the team to your organization, they will simply make assumptions about any missing or unclear requirements. This will almost certainly lead to a misunderstanding. Sometimes those might be small problems that are easily corrected once the build is delivered. But small problems can snowball into big ones, and that could lead to software that does not do what you expect it to.

One of the hardest parts of working with your project team is correcting problems. Keeping a team motivated when they are not performing as well as you would like them to is difficult enough when they are in-house (see [Chapter 10](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch10.html)). It is even more difficult to handle this situation in an outsourced environment. Not only is the team in another organization, where you don't have the same access to them, but they also do not have the same goals as you do.

Expecting your project team to give you credibility in reviewing their work is like expecting your car mechanics to listen to your criticisms of how they fix your car. Most of the clients that your team has dealt with in the past have probably been relatively hands-off. The clients have been mostly ignorant of how software is built at their organization, so it's likely that some of your team members have never even been addressed directly by a project manager at a client. Now they're faced with a project manager who is taking the time to evaluate their work in detail. This new attention may not necessarily be welcome at first, especially since you will almost certainly ask them to make changes to how they work.

You need to be aware of the fact that when you are saying something negative, you need to present it with a lot of objective supporting evidence. Over time, you will gain credibility with them. But, at least initially, you will have to prove that you know what you are talking about. You will not be recognized as an authority just because you are paying the bills. This is a reasonable attitude for the team members to take, since it's true of most of their clients. You have to give them good reason to understand that you are different.

The way to build up credibility with your team is to show that your interests are in line with theirs, and that you are often right. It's no coincidence that this is exactly what you need to do with your team in your own organization. The difference is that when you are a project manager of an in-house project, you come to the table already in an authoritative role. With this team, you have a much different role. You have wide authority to modify the team and to add or remove people, but you start out with very little credibility to make technical decisions (although you do have the authority to do so).

It's very important to see this from the team members' point of view. Consider a project in which the project manager is trying to guide the team toward making technical decisions that they are not fully comfortable with. In an in-house situation, if this situation leads to a failed project, the team can stand behind their manager and point to a decision that was made for them. With an outsourced project, the project manager is accountable in his own organization, but the team members are just as accountable in the vendor's organization. If the vendor loses the contract, the team members will be blamed and their careers will be impacted. So it takes some work to get them to trust you—and rightfully so.

It's harder to earn the trust of the vendor team than it is to earn the trust of an in-house team. You are not at the vendor every day, and the team does not know you and does not have the opportunity to get to know you. This is especially difficult when you make an unpopular decision, whether it concerns team responsibilities, technology, specific approaches to problems being solved, or some other decision. Every group in every organization discusses decisions that are made by managers; usually you will be there to defend those decisions if they are misunderstood. However, in an outsourced situation, your decisions have legs. People may attribute motives to your decisions that you did not intend. They may see those decisions as personal rather than professional. And worse, they may not feel comfortable enough talking to you to ask you about them. Team members have spent their careers at the organization learning its particular culture and processes, and the client is suddenly making changes to them. Often, you have not had time to evaluate how your changes will work with the process: this is another barrier to earning the team's trust, and you need to take it into account when working with them. The more open you can be to the team members' perspectives, the more intelligent your decisions will be. (Don't forget that you generally benefit from team members' experiences on past projects!)

You cannot expect to be at the vendor every time your decisions are called into question. This is why your partnership with the management of the vendor's company is so essential. They are your ambassadors to the team, and, if they trust you and understand why you have made your decisions, you can depend on them to smooth out these problems. This is why you need to be transparent with them about your decisions and the reasoning behind those decisions.

**Collaborate with the Vendor**

If you are the project manager on an outsourced project, your day-to-day work will be similar to what you would do on a project for software being developed in-house. There are, however, some important changes you need to make, in order to work with the vendor. Because it's so easy for the vendor to get lost in the details and lose context, tools and techniques in every phase of the software project must be modified, in order to keep the vendor in the loop and communicate your high-level goals for the product.

**Plan and Manage the Project Scope**

In an in-house project, you start with the project scope and the set of resources already known to the organization, and use those to estimate the schedule, budget, and due date (using the tools and techniques in [Chapters 2](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch02.html) through [4](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch04.html)). An outsourced project, on the other hand, is exactly the opposite: you start with a scope and a budget, and the vendor provides an estimate on the number of resources and the time expected to complete the project.

This is one of the main advantages to outsourcing: you have much more flexibility in allocation of resources. You can specify the scope and the expected budget, and ask for an estimate on both the number of resources and the expected time to complete the project. Alternately, you can specify the scope and the deadline, and ask the outsourcing vendor to estimate the number of resources and the project cost. However, in all cases, you will still need to know the scope of the project. Luckily, there is very little difference in defining the scope of an outsourced project and defining one within your organization. The description of the vision and scope document in [Chapter 2](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch02.html) can be used in an outsourced context as well.

However, there is one aspect of the project that is known during the planning stage but that is not covered in a typical vision and scope document for an in-house project: *knowledge transfer*. The outsourced vendor will not have prior knowledge of your organization, its products, or its users. It is your responsibility to teach your team about your organization and its goals. This can be done through meetings, documentation, working off-site at the vendor, bringing consultants from the vendor on-site to work with your organization's management and experts, or some combination of these things.

The main advantage to having someone from the vendor come to work at your organization is that it is less disruptive to you. It can be a successful way of transferring information; the person who works with you can act as your advocate within the project team, and she can use her understanding of your needs to keep the project team on track. The disadvantage is that you are depending on that person to collect the information, rather than teaching it yourself. This means that any misunderstanding might not be easily recognized early in the project. Also, each piece of information must first be learned by the vendor's representative and then retaught to all of the other members of the team; that takes time, and it may be less reliable than teaching it yourself.

The most effective way to avoid these problems is by working directly with the vendor's team. (Many vendors are located in the same city as the organizations they work for, but if your team is in another city or country, this could mean spending weeks, or even months, living out of a suitcase!) By going directly to the team, you can be sure that you have communicated your needs effectively. You can verify the team's knowledge in conversation and, if necessary, through a test or quiz. When a project manager works directly with the vendor's team, the knowledge transfer takes less time and there is less chance for misunderstanding.

Your method for knowledge transfer should be decided and covered in the vision and scope document. One effective way to do this is to include the vendor as a project stakeholder. This should make sense: the vendor has clear needs (including knowledge transfer) that must be fulfilled on the project. By adding the vendor as a stakeholder, you ensure that those needs will be considered when planning the project tasks.

Some project managers limit their interaction with the team to a set of milestone reviews during the project. For those managers, deadlines are the only monitoring tools. This is a mistake—the project manager cannot be disconnected from the project like that. It takes time and effort to work with the individual team members to verify that the work is being done properly, and that it meets the standards that your organization needs; if you don't take the time to do this, you can lose control of your project and not even know it. Unfortunately, by default, most outsourcing projects are set up so that the project manager at the client has a very small role in the project, and the contractor is responsible for keeping the team on track.

Success for the project manager and success for the vendor are often two different things. The project manager wants to deliver the software that meets the needs of the stakeholders in the project. The subcontracted team is trying to meet its contractual obligations. If the contract is not written specifically enough, or if the project manager is not able to collaborate with the team and revise its goals when necessary, it is likely that the goals of the project manager and the goals of the subcontracted team will diverge and possibly even be in conflict with one another.

**Do Your Own Estimation**

The estimation process in [Chapter 2](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch02.html) was based on the assumption that the work being estimated would be done by a known project team. That isn't always the case when you work with an outsourced vendor. Once your team is in place, you can use the process outlined in the chapter in much the same way. But when you are negotiating your initial contract, the people who are estimating the effort are most likely not the people who will eventually do the work.

Learn about your resources: ask them about their backgrounds and about what they are capable of, in order to understand who should be assigned to various tasks. Find out who is more senior in your team; adjust expectations, if necessary, based on the new estimates they give you. Remember, even if they don't work for your organizations, these are not just faceless resources—they're people with different capabilities and skill sets. The more effort you put into understanding them, the more likely they will understand your project and its goals, and the more accurate your team's estimates will be.

Once your team has been created, you should work with them; gather the team members together and hold a Wideband Delphi estimation session. (Alternately, use another method to gather estimates from the team—the important thing is that the team uses a repeatable method for estimating the software, and that they agree that the estimates are realistic.) When the team makes the new estimate, use the same materials that were used to create the initial contract. This is a your first chance to see if your original expectations were realistic.

It's important to keep in mind that the original estimate done by the vendor may already have been written into a contract. If the numbers that your team comes up with are different than those written into the contract, you may have a problem with your budget or with your own legal people. You may need to renegotiate the contract, or you may need to add (and pay for) more resources, in order to meet your deadlines. But it's much better to know this at the beginning of the project rather than find out later on, when the work is underway.

One common and very unfortunate pitfall that many people fall into is assuming that all of the estimates from a vendor are padded (just like an in-house project!). Even worse, many project managers assume that the vendors' estimates are chronically underestimated ("They said they only need two weeks for this task, but they'll really take five"). This is a mistake. If you consistently mistrust the estimates coming from the group, the people making those estimates will very quickly catch on and begin to meet your expectations. If your team really does have a problem estimating, that's a problem that should be dealt with and corrected through tracking the schedulevariance and other metrics in the same way you would track any employee on your team who has such problems (see [Chapter 4](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch04.html)).

**Maintain Your Own Project Schedule**

Giving up control of the schedule is a common mistake. It allows project managers, who are responsible for the ultimate success or failure of their project, to maintain almost no knowledge of how it is progressing or of who is doing what. It is your job to know why things are slipping, and whether or not commitments will be met—and you can't expect to adequately understand the complexities of your project with just a couple of status meetings. You must be an active participant in gathering knowledge about your project. In order to find out how the work is progressing and understand the problems your project is facing—which are necessary in order to make informed suggestions—you need to maintain your own project schedule.

Many project managers take a hands-off approach toward managing their outsourced projects , when they would never take such a risk with projects developed in-house. And it really is a huge risk; it means giving up a lot of control. Organizations of all types stretch the truth to keep their clients happy, and outsourced vendors are no exception. Keeping control of your project means verifying the status and the quality of the work product in the exact same way you would on your in-house project team. Review the work through both formal inspections and informal peer reviews (see the following section) to maintain an active understanding of your project tasks and their progress. But above all, know who is doing what and how far they have progressed.

If your project was done in-house, you would never let anyone keep the schedule for you. The project schedule should be kept and updated by you just as you would keep it in your organization. You should be notified immediately of any slippage, and hold regular schedule reviews as well as event-driven reviews with your outsourced project team. Everything in [Chapter 4](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch04.html) applies to your outsourced project team, and it is your responsibility to stay on top of it. If you don't, the team will keep moving forward on their schedule, perhaps toward their own goals, and will continue to bill you for their time, whether or not it meets your needs.

**Hold Reviews and Inspections**

A review is one of the most important tools a project manager has for knowledge transfer, and it is difficult to overstate the importance of reviews and inspections in an outsourced project.

The more feedback you give, the more the team will understand what you want. One of the most common causes for outsourced project failure is that the project manager does not check the team's work until major milestones are delivered. If the team misunderstands a major work product and it is not inspected until the end of a project phase, the effort for that entire phase could be wasted. Many of the most serious problems that plague outsourced projects can be caught early with inspection and constant collaboration.

Unfortunately, reviews in outsourced projects can be highly time consuming; much more so, in fact, than in an in-house project. In an in-house project, the team is already familiar with that particular organization's standards, and there are usually plenty of examples to work from. The project manager doesn't need to spend nearly as much time making sure that the team understands the work being accomplished. What's more, an in-house team normally understands the mission of the organization and the needs of its users. Many project managers take this for granted, and don't think to communicate these things to the vendor. It requires constant effort and vigilance on the part of the project manager to make sure that the needs are properly understood when moving work outside the organization.

In addition to knowledge transfer, reviews are also important tools for collaboration. It is important to encourage collaboration between the project team members at the vendor and the team members within the organization. When an inspection team is made up of people from both organizations, the only way for them to reach consensus on a work product, in order to approve it, is to collaborate on identifying and fixing the defects in that work product. After the inspection, everyone has a better understanding of the work to be done, as well as of how everyone else thinks about that work.

At the outset of the project, you must figure out which work products need to be inspected. You should add these inspections to the schedule as strategic milestones, to ensure that the vendor is in the loop. However, it will often be too time consuming to inspect every work product that you would in an in-house project, because inspections that span multiple organizations are much more effort-intensive than inspections that only involve people from a single organization. It will take some practice and experimentation before you find the "sweet spot" where you are catching enough defects, encouraging sufficient collaboration, transferring enough knowledge, and, most importantly, giving the proper guidance to your vendor team.

The most direct route for identifying and fixing defects is to have everyone on the inspection team meet in person. However, there are times (such as when an outsourced vendor team is in another country) when this is simply unfeasible. Luckily, there is a highly successful precedent in the software industry of collaboration without face-to-face meetings. Some of the most successful open source projects (such as Linux, Apache, Mozilla, Perl, PostgreSQL, and Subversion) are excellent examples of distributed teams who review each others' work without having face-to-face meetings. Project teams for most large, successful open source projects accomplish this through discussion groups, mailing lists, and other collaboration tools. They also browse log messages from the version control system, and use an automated project monitoring system (see [Chapter 7](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch07.html)) to keep the team up-to-date on the health of the code. Using these tools, project team members can collaborate on resolving the defects without having to meet face-to-face, but also without requiring enormous effort from the moderator. People collaborating in this way have produced some of the most reliable, defect-free software available at the time of this writing. You can take advantage of this in your own software projects.

[Table 11-1](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch11s03.html#appliedprojectmgmt-CHP-11-TABLE-1) shows an inspection process that has been modified to be used with an outsourced project. This script differs from the one in [Chapter 5](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch05.html) in that it does not require an inspection meeting. Instead, the inspectors prepare comments and send them back to the moderator, who consolidates them and works with individual inspectors to identify solutions that they all agree on. This requires much more time than a single inspection meeting because instead of having one single discussion about each defect, the moderator must have many different discussions with individual inspectors regarding each defect. It also requires that the selected moderator have extensive familiarity and expertise with the work product being inspected. This may mean that the project manager must serve as the moderator, but that's not always the case.

*Table 11-1. Inspection script for multiple organizations*

| **Name** | **Inspection script for use in multiple organizations** |
| --- | --- |
| Purpose | To run a moderated inspection (without a meeting) for a team with members in different organizations |
| Summary | In an inspection, a moderator leads a team of reviewers in reviewing a work product and fixing any defects that are found. The inspectors are from multiple organizations, so they never meet face to face. |
| Work Products | *Input*  Work product being inspected  *Output*  Inspection log |
| Entry Criteria | A moderator must be selected, as well as team of 3 to 10 people. A work product must be selected, and each team member has read it individually and identified all wording that must be changed or clarified before he or she will approve the work product. A unique version number has been assigned to the work product. |
| Basic Course of Events | 1. *Preparation*. The moderator distributes a printed or electronic version of the work product (with line numbers) to each inspector, along with a checklist to aid in the review. Each inspector reads the work product and identifies any defects that must be resolved, compiles those defects into a single document, and returns it to the moderator. 2. *Compile the draft inspection log*. Each list of defects returned by each inspector must be compared with the others, in order to identify and combine overlapping defects. The moderator compiles a draft of the inspection log that includes all distinct defects found by inspectors. The log does not yet contain any solutions to those defects. 3. *Identify conflicts*. The moderator searches for any defects reported by different inspectors that contradict each other. For each set of conflicting defects, the moderator holds a discussion (either in person, via teleconference or video conference, or using a collaboration tool like a mailing list or instant message system) between the inspectors who identified those defects, in order to identify the assumptions behind the defects and resolve them into a single defect. The inspection log is updated to reflect the combined defects. 4. *Identify solutions*. The moderator uses the same means to meet with individual inspectors, to identify solutions to the defects and add those solutions to the inspection log. If more than one person identified the same defect, they must all be involved in creating the solution. Inspectors may also identify additional defects that were not originally found, as well as their solutions. 5. *Compile and distribute inspection log*. The moderator compiles all solutions identified in Step 4 into the inspection log. Any defects that were not resolved are left as open issues to be resolved by the author. The moderator sends the final inspection log to all inspectors for confirmation. When the inspectors have confirmed that the log is correct, it is sent to the author of the work product. 6. *Rework*. The author repairs the defects identified in the inspection meeting. 7. *Follow-up*. Inspection team members verify that the defects were repaired. 8. *Approval*. The inspection team approves the work product. |
| Alternative Paths | 1. During Step 5, if one or more team members find errors in the inspection log, the moderator must address those errors before rework can occur. The script returns to Step 2. |
| Exit Criteria | The work product has been approved. |

While not every work product can be inspected, every work product in your project should be reviewed. Use deskchecks to have people in your organization spot-check the work done by vendors. This will help to find errors early on, and it can also generate confidence within your project team. If you are on top of the errors in your project, you can quickly correct problems if things start to go off track. There should be no surprises when delivery time comes.

In addition to inspecting work products, it is important to encourage the team within your organization to mentor their counterparts in the outsource vendor (and vice versa!). The more the two teams learn from each other, the less they will make mistakes or blame one another when things go wrong. The project manager of an outsourced project should try to create a cohesive team for the project that spans both organizations. While your team members may be paid by different organizations, they are all working toward the same goals and should feel comfortable both criticizing and praising each other's work.

**Maintain Control of Design and Programming**

The challenge of managing the design and programming of an outsourced project is collaborating with your engineering team. Clearly, the programming work is going to be taking place in the vendor's organization; the only question is whether the various aspects of the design take place in your own organization or at the vendor.

There are always design constraints for any piece of software: it may need to work with an existing enterprise infrastructure; you may have legacy code that must be integrated; you or others in your organization may have already decided on a language or platform; there may be user interface or other design standards that are already in place and must be met, etc. Minimally, you must make sure that these are communicated to the vendor.

The easiest way to maintain control over the design of the software is to design it yourself. Sometimes it's easiest to come up with a solution yourself that meets all of your needs, and to provide that as technical direction to the programming team at the vendor. You can do this by writing a technical specification or approach document. If you do this, it's very important that you put checkpoints in place, in order to make sure that the work is really being done according to these documents. The vendor team should inspect the document, and there should be periodic reviews or walkthroughs scheduled throughout the project, in order to verify that the work is being done according to your design.

One of the most common mistakes that project managers make on outsourced projects is to blindly trust that the vendor team fully understands and has properly interpreted all of the design documentation provided. If you do not check in at many points along the way to make sure that your intentions are being properly interpreted, it's very likely that you will end up getting software that technically complies with the specifications but that does not really meet your needs. This is a very frustrating position for a project manager. However, by implementing reviews and inspections (see above), you can avoid this problem.

Once you have put in place controls to ensure that the software design meets your organization's needs, you then need to ensure that the software is being built well. The tools and techniques in [Chapter 7](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch07.html) are especially helpful here. Require that the team build unit tests for all code, and that those unit tests are run regularly, with the results of the unit tests published across the teams. Use an automated project monitoring system to create nightly builds and run unit tests, and have the results published across teams. Have in-house software engineers do spot-check code reviews with vendor team members—preferably using code samples that were already reviewed by the vendor team. (To some people, it may seem like a waste of time to review code that has already been reviewed. In fact, it's especially helpful, because it allows your team to both verify that the code is being built properly and to audit the vendor's code review process.) Use automated code analyzers to enforce widely accepted coding best practices that may slip through a code review.

Some project managers may be hesitant to get this involved with the way the vendor team builds the code. It may seem somehow intrusive that you are second-guessing the abilities of the programmers at the vendor. It's important not to fall into this trap. Almost every vendor will welcome this collaboration. It means that you are sharing responsibility for the quality of the code. You are recognizing the challenges that the team developing the product will face, and you are helping them solve those problems. And it means, ultimately, that there will be no surprises: you won't end up with interminable QA cycles. In other words, both the project manager and the vendor should see this as a win-win situation.

**Take Responsibility for Quality**

You are responsible for the quality of your products, even when you contract the work out. It's easy to forget this. If you are paying a vendor to build a project, it somehow makes it easy to consider quality just another deliverable. It's not—it's a responsibility.

Many times, outside providers will offer "testing" of their products as part of the development estimate. It's not uncommon for this testing to be done not by an independent testing team but by the programmers who wrote it, or by junior members of the programming team with little or no test experience. When this is the case, they generally do not have much test documentation. Do not let this happen to your project. Demand resources allocated exclusively to test activities. Expect to review and inspect all test documentation. Be sure that your test activities are defined in your vision and scope document and, if possible, discussed in your contract.

Just as you would have members of your test team involved in all document inspections in your own organization, schedule your designated test team outside to review all documents and schedule their time researching and writing test plans as well. If you use specific metrics to measure product quality, expect your test team off-site to provide the same data and meet the same standards. Collaborate with them as you would your own team—spot-check the test plans, and test the results to be sure that the testing is being done as you would expect. Defect tracking should be done within your own organization. Test results should be monitored, and redundant tests should be done within your organization to be sure that bugs aren't slipping through.

Don't make decisions that undercut your QA team. When the project has reached its testing phase, there are many points when the project manager has to decide between doing things in less time and doing them thoroughly. As the end date looms closer and closer, it becomes less and less appealing to choose the "thorough" option.

For example, say you have a project 25% through its regression test cycle after a minor bug fix. Everyone expected the test to go well, but instead, a major defect was found. The lead tester at the vendor asks whether they should cut a new build now or if they should finish the regression test. It's possible that there are no more major bugs lurking within the code, and you could potentially cut out 75% of the time it takes to regress the software. Do you do it? If you are taking responsibility for the quality of the product, the answer is "absolutely not": you know that chances you will find another defect when you cut the next build are greater if you haven't completed the current regression test. But it's difficult to make this decision with looming deadlines, clients putting pressure on you, and, most importantly, a compliant vendor willing to cut out work in order to meet your deadline.

It's even harder to keep your commitment to software quality when a small change is made to a product that is already in production and rolled out to clients. It's very tempting to "just make the change" and run a cursory test (for example, testing only the fix itself or smoke testing only a few "core" areas of the software). If you stop to think about it, however, it's even more important to run a full regression test when making even a small change to a product that's already been rolled out. The users already have a good feeling about the software and expect it to keep meeting their needs. It's one thing to deliver a poor quality product from the beginning; it's quite another to replace software that's already working with software that's buggy. There's no easier way to upset your users and make your team look incompetent.

The point is that if you want to release software that will satisfy your users and stakeholders, then your off-site team needs your support, and your commitment to quality must be unwavering. If time is a real constraint, you should use proper planning. People in your organization must review and understand the test approach the vendor will take. Don't ever commit to deadlines that don't include estimates for software testing. (And always assume there will be multiple iterations of testing—it's foolish to assume there will be no defects and the first test will be the last!) If the estimated project plan runs past the deadline, cut the scope—not just the testing. Work with your QA team to identify time-consuming activities that can be automated or made more efficient.

Finally, don't just assume that just because the vendor's organization meets certain certifications or has been assessed (CMM Level 5, ISO 9000, Six Sigma—see [Chapter 12](https://learning.oreilly.com/library/view/applied-software-project/0596009488/ch12.html)), it means that they know better than you do how to run your project. Don't assume that the vendor never cuts corners. A good track record for past projects does not necessarily translate to a similar performance on your project. Even when you are working with a certified vendor, you still need to take responsibility for your portion of the work.