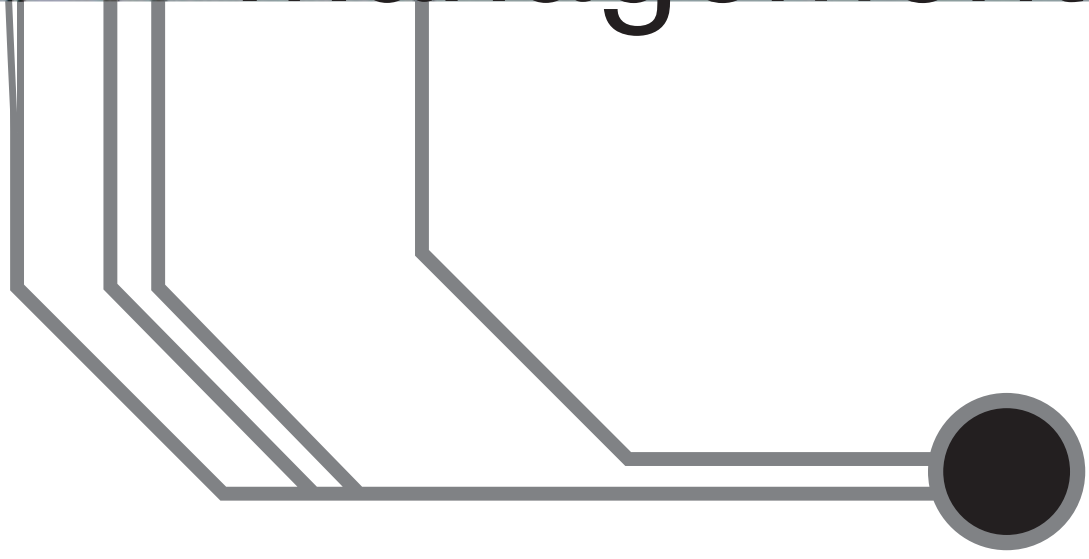
/Theory/In/Practice/

The Art of Project Management



O'REILLY®

Scott Berkun

CHAPTER THREE

How to figure out what to do

Few people agree on how to plan projects. Often, much of the time spent during planning is getting people to agree on how the planning should be done. I think people obsess about planning because it's the point of contact for many different roles in any organization. When major decisions are at stake that will affect people for months or years, everyone has the motivation to get involved. There is excitement and new energy but also the fear that if action isn't taken, opportunities will be lost. This combination makes it all too easy for people to assume that their own view of the world is the most useful. Or worse, that it is the only view of the world worth considering and using in the project-planning process.

> **"The hardest single part of building a software system is deciding what to build. No other part of the conceptual work is as difficult in establishing the detailed technical requirements, including the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the results if done wrong. No other part is more difficult to rectify later. Therefore, the most important function that the software builder performs for the client is the iterative extraction and refinement of the product requirements."**
>
> *—Fred Brooks*

It's not surprising then that the planning-related books in the corner of my office disagree heavily with each other. Some focus on business strategy, others on engineering and scheduling processes (the traditional focus of project planning), and a few on understanding and designing for customers. But more distressing than their disagreements is that these books fail to acknowledge that other approaches even exist. This is odd because none of these perspectives—business, technology, customer—can ever exist without the others. More so, I'm convinced that success in project planning occurs at the intersections in these different points of view. Any manager who can see those intersections has a large advantage over those who can't.

So, this chapter is about approaching the planning process and obtaining a view of planning that has the highest odds of leading to success. First I need to clarify some vocabulary and

concepts that different planning strategies use (it's dry stuff, but we'll need it for the fun chapters that follow). When that is out of the way, I'll define and integrate these three different views, explore the questions good planning processes answer, and discuss how to approach the daily work to make planning happen. The following chapters will go into more detail on specific deliverables, such as vision documents (Chapter 4) and specifications (Chapter 7).

Software planning demystified

A small, one-man project for an internal web site doesn't require the same planning process as a 300-person, $10 million project for a fault-tolerant operating system. Generally, the more people and complexity you're dealing with, the more planning structure you need. However, even simple, one-man projects benefit from plans. They provide an opportunity to review decisions, expose assumptions, and clarify agreements between people and organizations. Plans act as a forcing function against all kinds of stupidity because they demand that important issues be resolved while there is time to consider other options. As Abraham Lincoln said, "If I had six hours to cut down a tree, I'd spend four hours sharpening the axe," which I take to mean that smart preparation minimizes work.

Project planning involves answering two questions. Answering the first question, "What do we need to do?" is generally called requirements gathering. Answering the second question, "How will we do it?" is called designing or specifying (see Figure 3-1). A requirement is a carefully written description of a criterion that the work is expected to satisfy. (For example, a requirement for cooking a meal might be to make inexpensive food that is tasty and nutritious.) Good requirements are easy to understand and hard to misinterpret. There may be different ways to design something to fulfill a requirement, but it should be easy to recognize whether the requirement has been met when looking at a finished piece of work. A specification is simply a plan for building something that will satisfy the requirements.

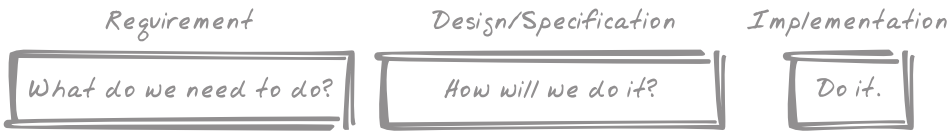


FIGURE 3-1. *An insanely simple but handy view of planning. If you don't know what you need to do, it's too early to figure out how to do it.*

These three activities—requirements gathering, designing/specifying, and implementing—are deep subjects and worthy of their own books (see the Annotated Bibliography). I'll cover the first two from a project-level perspective in the next few chapters, and implementation will be the focus later on in the book (Chapters 14 and 15).

Different types of projects

Several criteria change the nature of how requirements and design work are done. I'll use three simple and diverse project examples to illustrate these criteria:[1]

- **Solo-superman.** In the simplest project, only one person is involved. From writing code to marketing to business planning to making his own lunch, he does everything himself and is his own source of funding.
- **Small contract team.** A firm of 5 or 10 programmers and 1 manager is hired by a client to build a web site or software application. They draft a contract that defines their commitments to each other. When the contract ends, the relationship ends, unless a new contract/project is started.
- **Big staff team.** A 100-person team employed by a corporation begins work on a new version of something. It might be a product sold to the public (a.k.a. shrink-wrap) or something used internally (internalware).

These three project types differ in team size, organizational structure, and authority relationships, and the differences among them establish important distinctions for how they should be managed. So, while your project might not exactly match these examples, they will be useful reference points in the following sections.

How organizations impact planning

With the three project types in mind, we can examine the basic criteria for project planning. At any time in a project, there are basic questions that everyone should know the answers to. You might not always like the answers, but you and your team should know what they are. Most planning frustrations occur when there's disagreement or ignorance about these issues.

- **Who has requirements authority?** Someone has to define the requirements and get them approved by the necessary parties (client or VP). In the solo-superman case, this is easy: superman will have all of the authority he wants. On a contract team, there will be a client who wants strong control over the requirements and possibly the design. Lastly, a big staff team may have committees or other divisions in the corporation who will need to be served by the work (and whose approval in some way is required). There may be different people with high-level requirements authority ("It will be a sports truck") and low-level requirements authority ("It will get 20 mpg and have 4-wheel drive").
- **Who has design authority?** Similar to requirements, someone has to define the design of the work itself. The design is different from the requirements because there are always many different possible designs to fulfill a set of requirements. Designs, also like requirements, are often negotiated between two or more parties. One person or team might be responsible for driving the design process and developing ideas (designer), and another team provides guidance and feedback on the first party's work (VP). Note that because design skill is distributed in the universe independent of political power, people granted design authority might not be people with much design talent.
- **Who has technical authority?** Technical authority is defined by who gets to choose which engineering approaches are used, including programming languages, development tools, and technical architecture. Many of these decisions can impact requirements, design, and budget. The difference between technical decisions and design decisions is subtle: how something behaves and looks often has a lot to do with

how it's constructed. In some organizations, technical authority supercedes requirements and design authority. In others, it is subservient to them. In the best organizations, there is a collaborative relationship between all the different kinds of authority.

- **Who has budget authority?** The ability to add or remove resources to a project can be independent from other kinds of authority. For example, in the contract team situation, the team might have the power to define the requirements and design, but they might need to return to the client each time they want more money or time.
- **How often will requirements and designs be reviewed, and how will adjustments be decided?** The answer depends heavily on previous questions. The more parties involved in requirements, design, and budgets, the more effort will need to be spent keeping them in sync during the project. As a rule of thumb: the less authority you have, the more diligent you need to be about reviewing and confirming decisions, as well as leading the way for adjustments.

Although I've identified different kinds of authority, it's possible for one person to possess several or all of them. However, most of the time, authority is distributed across team leaders. The more complex the distribution of authority is, the more planning effort you'll need to be effective. In Chapter 16, I'll cover how to deal with situations where you need more authority than you have. For now, it's enough to recognize that planning involves these different kinds of power.

Common planning deliverables

To communicate requirements, someone has to write them down. There are many ways to do this, and I'm not advocating any particular method. What matters most is that the right information has been captured, the right people can easily discuss it, and good commitments are made for what work should be done. If the way you document requirements does all this for you, great. If it doesn't, then look for a new method with these criteria in mind.

For reference purposes, I'll mention some of the common ways to document requirements and planning information. If nothing else, knowing the common lingo helps translate between the various methods used by different organizations. You'll find some teams document the requirements informally: "Oh, requirements…just go talk to Fred." Others have elaborate templates and review procedures that break these documents into insanely small (and possibly overlapping) pieces owned by different people.

- **Marketing requirements document (MRD).** This is the business or marketing team's analysis of the world. The goal is to explain what business opportunities exist and how a project can exploit those opportunities. In some organizations, this is a reference document to help decision makers in their thinking. In other organizations, it is the core of project definition and everything that follows derives strongly from it. MRDs help to define the "what" of a project.
- **Vision/scope document.** A vision document encapsulates all available thinking about what a project might be into a single composition. If an MRD exists, a vision document should inherit and refer heavily to it. A vision document defines the goals of a project, why they make sense, and what the high-level features, requirements, or dates for a project will be (see Chapter 4). Vision documents directly define the "what" of a project.
- **Specifications.** These capture what the end result of the work should be for one part of the project. Good specifications are born from a set of requirements. They are then developed through iterative design work (see Chapters 5 and 6), which may involve modifying/improving the requirements. Specs are complete when they provide a workable plan that engineering can use to fulfill requirements (how much detail they must have is entirely negotiable with engineering). Specifications should inherit heavily in spirit from vision documents. Specifications define the "how" of a project from a design and engineering perspective.

- **Work breakdown structure (WBS).** While a specification details the work to be done, a WBS defines how a team of engineers will go about doing it. What work will be done first? Who will do it? What are all of the individual pieces of work and how can we track them? A WBS can be very simple (a spreadsheet) or very complex (charts and tools), depending on the needs of the project. Chapters 7 and 13 will touch on WBS-type activities. WBS defines the "how" of a project from a team perspective.

Approaching plans: the three perspectives

You may have noticed how each of the deliverables mentioned earlier represents one of two perspectives on the project: business or engineering. On many projects, these two views compete with each other. This is a fundamental planning mistake. Planning should rarely be a binary, or either/or, experience. Instead, it should be an integration and synthesis of what everyone can contribute.

To make this happen, a project manager must recognize that each perspective contributes something unique that cannot be replaced by more of something else (i.e., no amount of marketing strategy will improve engineering proficiency, and vice versa). For good results, everyone involved in project planning must have a basic understanding of each perspective.

WARNING

The following coverage of planning is industrial strength. If you see questions or situations that don't apply because of the size of your team or scope of your project, feel free to skim or skip them. I don't expect that everything I cover here applies to any single project. However, I'm trying to provide value to you for not only this project, but also the next one and the one after that. There are many angles and questions here that will prove useful to you in the long run, even if some of it doesn't apply to what you're working on today.

The business perspective

The business view focuses on things that impact the profit and loss (P&L) accounting of an organization. This includes sales, profit, expenses, competition, and costs. Everyone should understand their P&L: it's what pays their salaries or their contracts. When engineering teams are unaware of how their business works, many decisions made by management will appear illogical or stupid. Thus, it's in the interest of whoever's responsible for business planning to help others understand their reasoning. In the tech sector, people with job titles like business analyst, marketing, business development, product planner, or senior manager represent the business perspective.

Some projects have multiple business perspectives. If you work for a firm contracted to build a database server, you have your firm's business interests to consider, as well as the business interests of the client you are serving (hopefully they are in line with each other). The intersection of these perspectives can get complicated; I'm going to keep it simple here and assume projects are of the big-staff variety. However, it should be easy to extrapolate the following questions to more complex situations.

A good business perspective means that the team has answers for the following questions:

- What unmet needs or desires do our customers have?
- What features or services might we provide that will meet those desires and needs?
- On what basis will customers purchase this product or service? What will motivate them to do so?
- What will it cost (people/resources)? Over what time period?
- What potential for revenue (or reduced organizational operating costs) does it have? Over what time period?
- What won't we build so that we can build this?
- Will it contribute to our long-term business strategy or protect other revenue-generating assets? (Even nonprofits or IT organizations have a business strategy: there are always bills to pay, revenue to obtain, or revenue-generating groups to support.)

- How will this help us match, outflank, or beat competitors?
- What are the market time windows that we should target for this project?

Those responsible for the business perspective take bold views of the importance of these questions. They believe that the answers represent the bottom line for the organization and should strongly influence project decisions.

However, the business view doesn't mean that all projects must be slaves to revenue. Instead, it evaluates projects based on their contributions to the business strategy. For example, a strategic project might be essential to the organization but never generate any revenue.

Marketing is not a dirty word

The most unfair criticism of business folks is that they are just "marketers," somewhat of a negative label in the tech sector. I think marketing gets a bad rap. In MBA terms, there are four *P*s that define marketing: product, price, placement, and promotion. Defining the product and price is a creative process. The goal is to develop a product idea—sold for a profit—that matches the needs of the targeted customer. Research, analysis, and creative work are necessary in order to succeed. Placement, the third *P*, regards how customers will obtain the product (through a web site? the supermarket? the trunk of Fred's car?).

Finally, promotion—what marketing is often stereotyped to mean—is how to spread the positive word about the product to influential people and potential customers. Surprisingly, promotion is a small part of a business analyst or product manager's time (maybe 10–20%). So, marketing plans define much more than what the ads will look like or what promotional deals will be made. Also, note that the four *P*s of marketing apply to almost anything. There is always a product (HR web site), a price (free), a placement (intranet), and a promotion (email) for it.

But when the business perspective is dealt with alone, it shows only one-third of what's needed. The quality of a product influences sales, but quality does not come from marketing.

Quality[2] comes from successfully designing and engineering something that satisfies real customer needs. A proposed business plan that centers itself on technological possibilities (rather than conjectures) will make for good business.

A project manager, who uses only one perspective and fails, might never understand what really went wrong. His tendency will be to work harder within the same perspective instead of widening the view.

The technology perspective

While I was studying computer science at Carnegie Mellon University, it was common to talk to professors and students about new products. We'd always focus on what components these new software products used and how they compared against what could have been. Value was implicitly defined as quality of engineering: how reliable and performant they were or how much of the latest technology they took advantage of. Generally, we thought everything sucked. Exceedingly few products stacked up to our critiques. We wondered why the marketplace was packed end to end with mediocrity and disappointment. We'd even invent geek conspiracy theories to explain the evil decisions, which we thought were made against engineering purity and thus made little or no sense to us. Often, we'd focus blame on the marketing departments of these companies[3] (not that many of us understood what marketers did). Even in my first few years in the industry, the same kinds of conversations took place again and again. Only then there was greater scrutiny because we were competing with many of the products or web sites that we talked about.

When we looked at the world, we saw technologies and their engineering merits only. We never understood why poorly engineered products sometimes sold very well or why well-engineered products sometimes failed to sell at all. We also noticed that engineering quality didn't always correlate with customer happiness. For these mysteries, we had two answers. First, it had something to do with the magic powers of evil marketing people. Second, we needed smarter customers. But we didn't think much about our conclusions. Instead, we went

back to writing code or finding other products to tear to shreds. I was able to see my view for what it was only after I'd listened to some smart marketers and some talented product designers.

The technology view places the greatest value on how things should be built. It's a construction and materials mindset. There is an aesthetic to it, but it's from the technology perspective, not from the customer's perspective. There is a bias toward the building of things, instead of understanding how, once created, those things will help the business or the customer. In the stereotypical engineering view, a database that satisfies the engineer's aesthetic is sufficient, even if no customer can figure out how to do anything with it, or it fails to meet its sales projections.

As critical as that last paragraph might sound, many important questions come from the technology view only:

- What does it (the project) need to do?
- How will it work? How will each of the components in it work?
- How will we build it? How will we verify that it works as it's supposed to?
- How reliable, efficient, extensible, and performant are the current systems or ones we are capable of building? Is there a gap between this and what the project requires?
- What technologies or architectures are readily available to us? Will we bet on any new technologies that will be available soon but are not available yet?
- What engineering processes and approaches are appropriate for this team and this project?
- What applicable knowledge and expertise do our people have? What won't they be working on to work on this project?
- How will we fill gaps in expertise? (Train/hire/learn/ignore and hope the gaps magically go away.)
- How much time will it take to build, at what level of quality?

The customer perspective

This is the most important of all three perspectives. Because the project is made to serve the customer (and perhaps serve the business, but only through serving the customer), it follows that the greatest energy should be spent on understanding who those customers are. This includes studying what the customers do all day, how they currently do it, and what changes or improvements would be valuable in helping them do what they do. Without this information, engineering and business are shooting in the dark.

But, sadly, the customer perspective is the weakest in many organizations. It generally receives the least staffing and budget support. There are fewer people in most organizations that have been trained in understanding and designing for customers than their business and technology counterparts. And even when customer experts are hired (such as user interface designers or usability engineers), they are often restricted to limited roles in the project decision-making process and are granted few requirements or little design authority.

In any case, the customer point of view is built from two different sources: requests and research. Requests are anything the customer explicitly asks for or complains about. This kind of information is valuable because the customer has the greatest motivation to identify these problems ("Yes, my computer explodes whenever I hit the spacebar"), but it is also problematic because, in most cases, customers are not designers. They often blur the distinction between problems that need to be solved and specific ways of solving them. They may explicitly ask for a feature, such as print preview, without describing the real problem (people throw away too much paper). If the project team can start by understanding the problem, there may be many ways to solve it that are cheaper or better than the feature requests. Even skilled designers often struggle at designing for themselves.[4]

There are two kinds of experts who understand customers and design for them: usability engineers and product designers. Usability engineers are experts in understanding how people work, and they provide metrics and research to help project

teams make good decisions from day one of project planning. Product designers, or interaction designers, are people trained in how to take that data and convert it into good designs for web sites or products. If your organization is fortunate enough to employ these fine folks, involve them early on. Ask them to be advocates for this point of view. If you're working without them, you are at a distinct disadvantage to your competitors. Consider hiring someone to consult and advise on where these efforts would be of the most value.

Without expert help, the project manager must make do on her own. This is possible, but because it's often the least interesting perspective for folks with engineering backgrounds and is least understood by senior management, it typically gets less support than the other points of view. Enough resources and seniority need to be invested in the customer perspective to balance out the technology and business ones. Otherwise, surprise: the customer perspective won't be credible and won't be heard.

The important questions from the customer view include:

- What do people actually do? (Not what we think they do or what they say they do.)
- What problems do they have trying to do these things? Where do they get stuck, confused, or frustrated?
- What do they need or want to do but aren't able to do at all?
- Where are the specific opportunities to make things easier, safer, faster, or more reliable for them?
- What design ideas for how to improve how the thing should work—in terms of what people actually do—have the most potential for improving the customer experience?
- How can those ideas be explored? What prototypes, sketches, or alternatives need to be investigated to help us understand the potential for the project?
- What core ideas and concepts should the project use to express information to users?

The magical interdisciplinary view

These three points of view always overlap each other. Every business consideration has technical and customer implications (which is the same for all of the other permutations). So, getting the best planning perspective requires laying out each view on equal footing and seeing where the similarities and differences are. Some decisions will need to be made that favor one perspective over another, but that shouldn't be done by accident. It should support an intelligent strategy derived from getting as much value from each perspective as possible.

By investing time in exploring all three perspectives, it's possible to see opportunities for smart strategic decisions. It might be possible to satisfy some of the top issues or goals from each of the three perspectives by defining a project targeted at where the three perspectives overlap. Those are areas that have the greatest potential value to the organization because one effort can simultaneously address business, technology, and customer goals.

Almost as important as its strategic planning value, using a Venn Diagram (like the one in Figure 3-2) can defuse perspective bias of engineers or marketers. It helps teams see overlapping points of view, rather than only competing ones. Early and often during project-planning discussions, this diagram or something like it (e.g., a diagram that includes a list of potential goals from each perspective) can be used to frame suggestions made by people who have bias toward one view of the project. When ideas are suggested, they can be mapped against this diagram to see how they contribute to all three perspectives. The PM plays a key role in making this happen, by proactively using his generalist nature to unify all three views into one.

One way to accomplish this is to establish early on that there will always be great technological ideas that do not benefit the business or the customer, as well as great ideas to help customers that are not viable for the business or possible with current technology. This gives everyone the power to identify one-dimensional ideas and call each other on them. It also generates respect across perspectives because everyone is forced

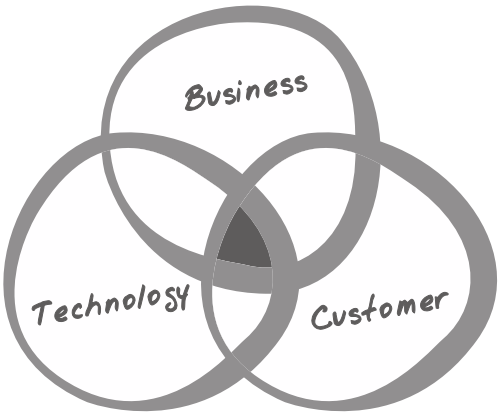


FIGURE 3-2. *The three perspectives.*

to realize that they need to collaborate with people who have knowledge they don't possess in order to be successful.

But if no effort is made to bring divergent points of view together, the conflicts are rarely addressed head on. Instead, project-planning meetings become battlefields for attacking and defending opinions based on these perspective lines (and not on the true merits of the ideas themselves). Often when I've consulted with project teams, the problem I was asked to help with had nothing to do with their ability to plan a project. Instead, there was an unresolved, or even unspoken, conflict of opinion about why one department—engineering or marketing, for example—is more important than the other. Their singular perspectives not only caused the problem but also made it impossible to see the cause of the problem.

Years ago, I was involved in one of these silly wars myself. I was the program manager for web-search features on Internet Explorer 4.0. Two business development people were assigned to us, and they were negotiating deals with the major search engines of the time (Excite, Yahoo!, Lycos, AltaVista, etc.). We argued with these business experts over design decisions, continually debating over what was best for the customer versus what was best for the business. We each believed that we held the authority (I spoke for the design/engineering staff, and they provided the business arguments). We argued on the same points for weeks, always debating the specific decisions and never stepping back to evaluate our hidden philosophies on what made for good products. Things got so bad that we brought in our group manager to help us reach a compromise.

I'm convinced a broader view of the world would have helped everyone involved. We were all so invested in our egos and beliefs that we were willing to spend tons of time fighting over tiny points, instead of working to understand all of the perspectives on what we were building. A better vision document could have helped, but that was impossible because the business challenges of the Internet were so new to the industry (circa 1997). However, had we been sharing each other's knowledge, instead of resisting it, we might have had a shot at finding a mutually beneficial compromise.

Bringing an interdisciplinary view to a project enables you to make choices that cut across the very boundaries that limit your competitors. It also gives you stronger arguments for any decision you choose to make. Instead of only claiming that a specific design will be easier to build, you can also say why marketing will find more opportunities to sell that design (provided, of course, that you're not just making up these claims). Sometimes, this will require you to make sacrifices. When you're looking for the best solutions, they won't always correspond to what you're good at doing, or which ideas you personally prefer. But if you're able to make those sacrifices, you gain the conviction and sincerity required to get others to do the same. You can then call others on favoring pet ideas over what's best for the project. People will get behind decisions they don't completely agree with if they see that an open mind, working in the interests of the project, is at work making those decisions.

The balance of power

If you work in a large organization, you should consider a certain political factor to balance the view of a project. I call this factor the power ratio. How is power on the project distributed across people who represent these three views? For example, if engineers outnumber business analysts by 3:1, the engineering view will tend to dominate decisions. The power ratio is simply the ratio of the number of people prone to a given view. To have a balanced perspective, the ratio should be 1:1:1 (engineering to business to customer). The natural power ratio is the raw count of people who have expertise in each view.

The more out of balance the ratio is, the larger the shift will be toward a given perspective.

But raw numbers of people don't define how much power they have. Napoleon's army had thousands of soldiers, but there was only one Napoleon. There may be 10 programmers and 1 marketer (10:1:0), but the marketer may have as much power over the project, given his role or seniority, as the others combined. This means a manager can compensate for any natural ratio by granting power to those who should have more influence on the project. And because the nature of a project changes over time, different perspectives should have more power at different times. Consider how you can delegate decisions (see Chapter 12) to find the right balance for the project at the right time.

Asking the right questions

The simplest way to frame planning work is to refine a set of questions that the planning work needs to answer. They should be pulled from the three perspectives with the intention of combining them into a single plan. Initially, they can be explored independently. Early project definition can be open ended. People can run with pet ideas or hunches for a while, they just need to be framed. Everyone should know that it will all come together into MRDs or vision documents, which will require many discussions that combine business, engineering, and customer thinking into a single plan.

The questions (often called project-planning questions) should be pulled from the three lists discussed earlier, based on their relevance to the project you're working on. If it's a new project (not a v2), then you'll need basic questions to define the fundamentals. If it's a small upgrade to an existing system, there may be fewer business and customer issues to consider. But no matter what the project is, do the exercise of running through the questions. It will force out assumptions and ideas that haven't been recognized and give everyone a starting point to discuss them.

This project-planning question list should be free of most perspective boundaries. Instead, you'll have a holistic point of view of the project, which can be divided, as needed, into engineering, business, or customer considerations. For example, the following list shows more complex versions of questions listed earlier:

- What are the three or four useful groupings we can use to discuss the different kinds of customers we have? (For example, for a word processor, it might be students, professionals, and home users. For an IT database, it might be sales, receptionists, and executives.) How do their needs and behaviors differ?
- What demographic information can help us understand who these customers are? (Age, income, type of company, profession, education, other products owned or web sites used, etc.)
- Which activities is each user group using our product for? How does this correspond to what they purchased the product for? How does this correspond to how we marketed the product? What problems do they have in using the product to satisfy their needs?
- Who are our potential new customers, and what features, scenarios, or types of products would we need to provide to make them customers? (What are the demographic profiles of these new customers?)
- Do we have the technology and expertise to create something that satisfies these needs and problems? (For each identified need, answers of yes, maybe, and no can often be sufficient, at least as a first pass.)
- Can we build the technology and obtain the expertise to create something that satisfies these needs and problems? (Yes, maybe, no.)
- Are there significant opportunities in a new product or line of products? Or are the needs tied directly to the current product or line of products?
- Are there viable business models for using our expertise and technology to solve these identified problems or needs? (Will profits outweigh costs on a predictable timeline?)

- What are the market timelines for the next release or product launch? Which windows of opportunity make the most sense to target?
- What are competitors in this marketplace doing? What do we think their strategies are, and how might we compete with them?

Answering the right questions

It can take hours or weeks to answer these questions, depending on the depth and quality of the answers needed, which is defined by the project manager or group leader. As a rule of thumb, the more strategic the project is expected to be, the more important the quality of this kind of definition and planning research is. For tactical projects that are directed at minor issues or short-term needs, less depth is needed. You might need to consider only a handful of questions, and you can base your answers largely on how you answered them for the last project. But for important projects, this information will be invaluable in any midproject adjustments or changes, not only in the planning phase.

Some of these questions are best answered by business analyst types, others are best answered by lead programmers or usability engineers. Often, the best answers come from discussions among these experts and the sharing of notes, sources, and opinions. It can be expensive and time consuming to do this work, but that's the nature of planning. Buying a house or car, moving to a new country, or writing a book requires significant planning efforts to make the process work out well. If you do it right, it enables sharper and quicker decision making throughout the rest of the project. (I'll talk more about this in Chapter 14.)

What if there's no time?

In the worst case, even if no research exists and no time is allocated for doing proper investigation, ask these questions anyway. Simply raising good questions invites two positive possibilities. First, intelligent guesses at the right question are better than nothing. A well-asked question focuses energy on

the right issues. Even if you only have time for guessing, speculation on the right issues is more valuable than speculation on the wrong issues. Second, the absence of research into core questions can raise a red flag for leaders and management. The long-term health of an organization is dependent on its ability to make good plans, and even though investments (hiring someone or providing funding) might come too late to help this project, it can definitely help the next one.

Catalog of common bad ways to decide what to do

There are always more bad ways to do something than good ways, and project planning is no exception. As an additional tool toward sorting out the good from the bad, Table 3-1 shows some of the lousy approaches I've seen used. I offer these in the hopes that it will help you recognize when this is going on, and why these approaches are problematic.

Bad way	Example	Why it happens	The problem
We will do what we did last time.	"Version 3.0 will be like 2.0, only better!"	Often there isn't the desire or resources to go back and do new research into the business, technology, and customer issues.	The world may have changed since v2.0. Without examining how well 2.0 did against its goals, the plan may be a disaster.
We'll do what we forgot to finish last time.	"The feature cuts for Version 2.0 will be the heart of 3.0!"	Items that were cut are arguably well understood and partially complete, making for easy places to start.	Remaindered features are nonessential. Focusing a release on them may not be the best use of resources.
We'll do what our competitor is doing.	"Our goal is to match Product X feature for feature."	It's the simplest marketing strategy. It satisfies the paranoid, insecure, and lazy. No analysis is required.	There may be stupid reasons a competitor is doing something.

TABLE 3-1. *Common bad ways to decide what to do*

Bad way	Example	Why it happens	The problem
We will build whatever is hot and trendy.	"Version 5.0 will be Java based, mobile-device ready, and RSS 4.0 compliant."	Trends are trends because they are easy and fun to follow. People get excited about the trend, and it can lend easy excitement for boring or ill-defined projects.	Revolutions are rare. Technological progress is overestimated in the short term, underestimated in the long term. Customer problems should trump trendy fads.
If we build it they will come.	"Project X will be the best search engine/web editor/widget/mousetrap ever."	By distracting everyone to the building, rather than the reason for building, people can sometimes avoid real planning.	Does the world need a better mousetrap? People come if what is built is useful to them, not because a team decided to build something.

TABLE 3-1. *Common bad ways to decide what to do (continued)*

The process of planning

In whatever time is allotted for defining the project, create a simple process for answering the planning questions. If possible, each perspective (business, technology, and customer) should have one person with expertise in that area driving the research of information, generating ideas and proposals, and reviewing her thoughts with peers from other perspectives. The trick is to keep this small enough to be productive, but large enough in perspective to be broad and comprehensive. A group of 10 people will be much less effective at discussing issues and developing team chemistry than a group of 5 (see Chapter 9).

From experience, I'd rather deal with the bruised egos of those who are not main contributors to planning than include too many people and suffer a year or longer on a poorly planned and heavily compromised project. The mature people who you do not include will understand your reasons if you take the time to explain them, and the immature will have an opportunity for growth, or motivation to find employment better suited to their egos.

If you're using planning deliverables like the ones I briefly described earlier in this chapter, the goal of the planning group should be to create and publish those documents for the team.

The planning phase (see Figure 3-3) ends only when those documents (or more importantly, the decisions they contain) are completed.

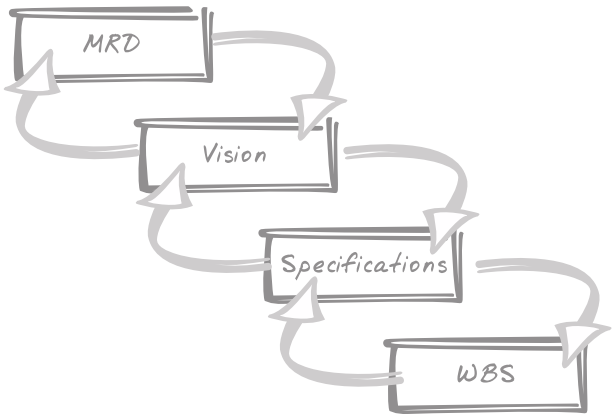


FIGURE 3-3. *The feedback between levels of planning.*

A draft version of each planning document should be prepared early enough to incorporate feedback from the team before a final version is due. As shown in Figure 3-3, there may even be a simple feedback loop between deliverables. When the draft of an MRD is created, someone may be able to start working on the vision document, raising new questions for the MRD that improve it before it's finalized. This pattern repeats through all of the planning work. So, even if there are hard deadlines for finishing planning docs, some overlap in time is healthy and improves the quality of the process. As shown in Figure 3-4, when a project is in mid-game (implementation), it becomes harder, though not impossible, for this kind of feedback to propagate back up the planning structure. (Alternatively, Figure 3-4 can be thought to represent a contracted team that has influence over specs and work assignments only.)

The daily work

As far as the daily work of planning is concerned, there's no magic way to go about doing these kinds of collaborative tasks. People are people, and it's impossible to skip past the time it takes to get individuals who are initially of different minds to come together, learn from each other, and make the arguments or compromises necessary to move things forward. There will be meetings and discussions, and probably the creation of email distribution lists or web sites, but no secret recipe of these things

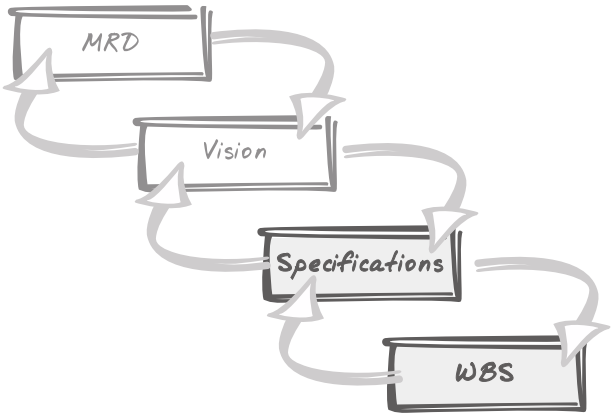


FIGURE 3-4. *As time goes by, it should become harder (though not impossible) for changes to propagate back up the planning structure.*

makes a big difference. Be as simple and direct as possible. The leader sets the tone by starting the conversations, asking the important questions, and making sure the right people are in the room at the right time. However, there are three things to keep in mind:

- **The most important part of the process is the roles that people are expected to play.** Who has requirements authority? Design? If many people are involved, how will decisions be made? How will ties be broken? With these sorts of relationship issues defined early on, many problems can be avoided or, more probably, handled with composure and timeliness. (See Chapter 10 for more on relationships and defining roles.)
- **Everyone should know what the intermediary points are.** What are the milestones between day one of the planning effort and the day when the project definition is supposed to be complete? The timeline for deliverables—such as reports, presentations, review meetings, or vision documents—should be listed early and ownership defined for each of them. When exactly does "planning" end and design or implementation begin? There should be good, published answers.
- **There should be frequent meetings where each perspective is discussed.** Reports of new information or thoughts should be presented, and new questions or conclusions should be raised. Experts from elsewhere in the organization or the team should be pulled into these meetings when they have expertise that can help, or if their opinions would be of value to the group.

The project manager is often responsible for consolidating each meeting and discussion down into key points and making sure conclusions reached are written in stone in a place the group can easily reference. Questions or issues raised should be assigned appropriately and then discussed at the next meeting.

Customer research and its abuses

There are many different ways to abuse information about customers. Simply claiming that customers are important doesn't signify much. It takes no work to say "We care about customers" or "Customer satisfaction is important" because rarely does anyone ask how those beliefs map to organizational behavior. Even though in the last decade much progress has been made in refining methods for researching and understanding customers, most of it has not penetrated through to management- or engineering-centric organizations. It's still uncommon for project teams to have an expert in customer research, interface design, or usability available to decision makers.

By far, the most prevalent mistake I've seen in customer research is over-reliance on a single research method as the source for decision making. The fundamental problem with all research, scientific or otherwise, is that a given study assesses only one point of view on an issue (we'll discuss this again in Chapter 8). Each method for examining something is good at measuring certain attributes and horrible at measuring others (see Table 3-2). Just as you would never use a speedometer to measure your weight, or your bank account to measure your blood pressure (though they may be related), there are some things that surveys and focus groups are good for and others that they are not.

Method	What is it?	Pros	Cons
Focus group	A group of potential customers are brought together to view prototypes and give opinions in a facilitated discussion.	Can get many opinions at once. Allows for extended suggestions and open dialog.	Discussions are difficult to analyze and easy to misinterpret. Poorly trained facilitators create deceptive data.[a]
Survey	A series of questions are given to potential customers.	Low-cost way to get information from large numbers of people. Good for very broad trends.	Information reliability is low.[b] Authoring surveys without biasing answers is difficult. Easy to misinterpret data.
Site visits	Experts or team members go to the customers' work sites and observe them doing their work.	Observe the true customer experience. Often this is the most memorable and powerful experience for the team.	The data is most valuable to those who did the visit: it's hard to transfer to others or to use quantitatively.
Usability study	Selected customers use a design in a controlled environment. Measurements are taken for how many scenarios they can complete, in how much time, and with how many errors.	Quantifies how easy it is to use anything. Provides evidence for specific problems. Most valuable when done early, before project begins.	Little direct value for business or technological questions. Can be wasted effort if done late or if engineering team doesn't watch often.
Market research	The market of the product is examined to see how many customers there are, what the competing products cost, and what the revenue projections are.	Only way to capture the business view of a market or industry.	Doesn't explain why products are successful, and it focuses on trends and spending, rather than people and their behaviors.

TABLE 3-2. *Common customer research methods*

[a] Focus groups tend to bias people toward being helpful. They don't want to insult their hosts, and they will often be more positive and generous in considering ideas than they would otherwise.

[b] Consider how diligent you were in answering questions in the last survey you took. If you never take surveys, ask yourself about the kinds of people likely to spend lots of time taking surveys.

Experts at customer research do two things: they choose the method based on the questions the project team needs to answer, and they make use of multiple methods to counteract the limitations and biases of individual approaches. Table 3-2 outlines some of the major research methods and their high-level tradeoffs.

As a program manager at Microsoft, on the best project teams I worked on, I had access to many of these sources of information. I'd often have to request answers to specific questions that went beyond what I was provided with, but there were dedicated experts in the organization who would generally do this for me. On other teams with less support, I'd have to go and make do on my own (typically with less success because I had many other things to do as well, and I wasn't as proficient at getting results as a full-time expert would be).

Even with no resources or budget, a few hours of work toward answering those planning questions can sometimes provide useful results. Focused energy spent on smart web searches and library inquiries (real librarians are often more powerful tools than web sites) can reveal sources that are infinitely more useful than nothing. Over time, the skills and experience in doing this kind of research will grow, and it can take less time in the future. More importantly, having done some of this kind of work on your own will put you in a more informed position to hire someone to do it for you, should the budget or headcount finally be offered to you.

With any source of data, skepticism and healthy scrutiny help refine and improve its value. Assumptions should be questioned, and known biases of different kinds of research should be called out at the same time the research is presented in a discussion. This doesn't mean that that data should be thrown out simply because there isn't enough of it or because there are valid questions about it. Instead, the team should try to look past the flaws to find the valuable parts that can be used to influence discussions and give a better perspective on what the reality of the customer's experience is like. No form of data is perfect: there are always biases, caveats, margins of error, and hidden details. The project manager has to be able to see past the biases and make intelligent use of what's available to make better decisions.

Bringing it all together: requirements

Planning creates large amounts of interesting information (asking many questions tends to make that happen). The challenge becomes how to simplify the information and convert it into a form useful for defining a plan of action. At a high level, a vision document is where all of the perspectives, research, and strategy are synthesized together. We'll talk more about that special document in the next chapter. But at a medium to low level, the simplest tool is the use of requirements. Vision documents often contain requirements information, but depending on whether specifications or other, more focused documents will be written, detailed requirements might be contained elsewhere.

Many projects use the requirements as the way to define the direction of a project. A requirement by definition is anything the team (and client) agrees will be satisfied when the project is completed. In the simplest sense, ordering a pepperoni pizza is an act of requirements definition. You are telling the pizza chef specifically what you want. He may ask you questions to clarify the requirement ("Do you want a soda with that?"), or he may negotiate the details of the requirement ("We're out of pepperoni, will you accept salami instead?"). In the more complex case of software development, good requirements are difficult to obtain. There are many different ways to interpret abstract ideas ("make it run fast" or "make it crash less often"), and the process of eliciting requirements can be difficult.

There are established methods for developing and documenting requirements, and I recommend familiarizing yourself with them (see the excellent *Exploring Requirements: Quality Before Design*, by Donald Gause and Gerald Weinberg, Dorset House, 1989). Depending on what authority you have over the requirements process, there are different ways to go about doing it so that you'll obtain good results. The details of these methods and how to apply them are out of the scope of this book. However, I can offer you one simple method that I think is easy to use and generally very effective: the problem statements method.

Problem statements are one- or two-sentence descriptions of specific end user or customer issues. They should be derived from any of the research that was performed or from specific customer requests. They should be written in a format that identifies a problem or need from the customer perspective (as opposed to the engineering or business perspective). This will ensure that the point of view of the impact on the customer is maintained and not unintentionally distorted by other perspectives. Problem statements also help avoid some of the common requirements mistakes that teams make (we'll cover them briefly in Chapter 5).

As an example, here's what a list of problem statements for an intranet web site might look like:

- It is hard to find commonly needed items on the home page.
- Pages with department information are very slow to load and users have to wait.
- The database query page crashes when working with large tables, and users have to start over with their work.
- The site does not provide automated access to HR services, which are time consuming to do manually.
- Search results are difficult to scan with the current layout.
- The registration page doesn't warn about required fields, and it's too easy to make mistakes.
- The status page doesn't include information about email, and users cannot find out why their email isn't working.
- There is no way to save preferences or options for how the home page is displayed.

Note that these are not bug reports. These issues may have never been identified as things the web site needed to do. Problem statements should be broader than and different in perspective from bugs because the idea is to capture what's missing from the customer's perspective, instead of only what is broken from a technical perspective.

Each of these one-sentence statements can be followed by supporting evidence or examples (say, screenshots from the web site or product that provides context for the issue, or references to the usability study or other research that surfaced the problem) to help tell the story and explain why and how the issue occurs (or why the omission of a kind of functionality is significant). But this supporting evidence should not mix with the problem statement itself, or with engineering plans or business objectives. For sanity, these customer problem statements should remain purely about customers and their needs.

Problems become scenarios

Because problem statements represent the current state of the world, a project needs something else to express how the world will be when the work is completed. For this purpose, problem statements need to be converted into what are called feature statements or scenarios. There are many different ways to do this; use-cases are one popular method (see Alistair Cockburn's *Writing Effective Use Cases*, Addison Wesley, 2000), but there are many others.

Each scenario is a short description of something a customer will be able to do as a result of the project, or the tasks they will no longer have to do because the project automates those tasks for them. The idea is to describe these things from the customer or user's perspective and to avoid any description of how these benefits will be achieved—that comes later. For now, what's important is that the team is able to articulate and discuss which scenarios have the most value. Considerations for the business value of solving each scenario or their technological feasibility should be reflected in how the scenarios are prioritized.

The feature statements themselves should become the way to most easily represent what's been learned about customers and what the project will be focused on providing for them. Based on the previous list of customer issues, here is what some feature statements might look like:

Possible features of Project X:

- Commonly used items will be easy to locate on the home page.
- Search results will be easy for most users to read quickly.
- The site will provide easy, automated access to HR services.
- The registration page will make it easy to enter information without mistakes.
- Department information pages will be at least as fast as the home page itself.
- The database query interface will be as reliable as other parts of the system.
- Users will be able to learn about email server status issues in a simple and convenient way.
- Users will have a convenient way for the system to remember their preferences.

Feature statements should never describe a specific solution or design, but should instead explain the solution's impact on the customer. (This is easier said than done. Most engineers and creative people love to solve problems. If you describe a problem, they'll want to jump right into solving it instead of spending time trying to elaborate on or refine the problem. It's common to require a temporary ban on solution proposals during discussions of problem lists and scenarios. Simply ask people to write down their ideas during the meeting, and then discuss them later. Make exceptions for ideas that completely eliminate problems from the lists or identify them as trivial.)

By postponing deep discussion about design alternatives, the team can focus on clarifying the real goals of the project. These feature statements can be ordered roughly by importance, helping to define the shape of what the project will be. If this is managed well, when the time comes to explore and define designs, it will go much faster because everyone will be working toward the same results (instead of being distracted by technologies or their favorite ideas for solutions). Because so much is riding on these short descriptions, they need to be

written carefully and with consideration for how long they'll be used by the project team. It often takes several passes and reviews to get them right, but once complete, they'll rarely need to be redefined over the course of a project.

Integrating business and technology requirements

With a list of potential features that grew out of user research, additional features to satisfy business or technology considerations can be added. But a primary question must be answered: what is the purpose of these additional requests if they do not contribute toward helping customers? Before adding new features, the existing list should be reviewed to see which ones already represent these business and technology considerations. This forces all discussion to be centered on customer impact and benefit, without prohibiting specific technology or business considerations.

It's entirely possible that business requirements to exploit certain market opportunities are represented by one or more features already on the list. Technology requirements should also be tied back to benefits that those engineering efforts will create for customers. Any business or technology requirements that don't connect with customer benefits (short or long term) should be scrutinized. These noncustomer-centric features should be carefully defined to make sure they do not negatively impact the customer's experience.

And even if marketing demands an addition that has no ties to improving the customer experience, everyone will know that this is the case and respond accordingly. Sometimes, it's necessary to add a feature to help sell a product, despite its dubious end-user value, or to satisfy a demanding client or executive. But by organizing the planning process first around customer research, problem statements, and resulting features, everyone will have to make arguments within that context. Warning bells should go off if the majority of features in a release have no direct connection to the customer. If they can be reviewed by their relationship to a customer-centric list, random or self-serving requests will stand out to everyone in

the room and demand additional debate and discussion. This gives the project manager every opportunity to define a level playing field of features that has the best interests of both the customer and the organization in mind.

Summary

- Different projects demand different approaches to planning.
- How planning is done is often determined by who has what authority. Requirements, design, and budget are the three kinds of project authority that impact planning.
- There are some common deliverables for planning projects: marketing requirements documents (MRDs), vision/scope documents, specifications, and work breakdown structures (WBSs).
- The most powerful way to plan a project involves use of three equal perspectives: business, technology, and customer. The customer perspective is often the most misunderstood and misused.
- Asking questions forces good thinking and directs planning energy effectively.
- The process of defining requirements is difficult, but there are good references for how to do it well.
- Problem statements and scenarios are a simple way to define and communicate requirements. They are easily converted into design ideas without losing clarity about what's important and what isn't.