Chapter 1
Planning Ahead

"Intangible results" is just another way of saying "poor planning."
—Anonymous

This chapter is intended for executives, decision makers, business analysts, and project leaders who need to make decisions, set expectations properly, and budget the time, money, and people for an SFDC project. Readers will learn how to make planning decisions that give them the best chance of success and the quickest possible failure recovery.

A truly effective Salesforce.com (SFDC) system can dramatically improve your business results in terms of customer satisfaction, executive decision making, smooth partner/channel interactions, and, of course, sales. But the benefits of increased visibility, streamlined operations, and faster cash-to-cash cycles are anything but automatic. The whole reason for this book is to guide you through a series of choices and actions to get you the best results with the least amount of cost, delay, and risk.

One of the key success factors in any sales force automation and customer relationship management (SFA/CRM) project is strong executive sponsorship that propels the implementation team and the end users. You'll want to check out Chapter 6 to learn about the best sponsors for your company's situation—but here we'll assume that you already have an executive with fire in his or her belly.

Peter Drucker wrote that the most important part of solving a business problem is a well-formed question. The same is true here: the best way to achieve results quickly is to clearly articulate and prioritize goals. This chapter focuses on understanding the requirements, schedules, and costs that are involved in completing any SFDC project successfully. You'll want to make sure the entire team understands the business drivers and the tradeoffs that will shape your SFDC implementation, its integration with surrounding systems, and the instantiation of business processes that will streamline the way your business runs.

Read the Map Before You Start the Journey

Salesforce provides a large number of helpful roadmaps, white papers, guidelines, and tips for implementation projects. This book builds on those basics, so you'll definitely want to access the Help & Training system inside SFDC itself, and read up on the materials in Salesforce's main Web site (http://blogs.salesforce.com/guides/), Successforce (http://success.salesforce.com/), its community site

(www.salesforce.com/community), and user-group sites. Check out www.SFDC-secrets.com for even more resources.

Developing a Model of Your Customer Relationship

SFA/CRM applications are designed to improve the speed of acquiring new customers and to achieve the highest amount of revenue or profits from accounts, thereby raising the customer lifetime value. Before you start detailed planning for any SFA/CRM project, you need to develop a model for how your organization works, showing the basic organizational ownership for each stage of the customer cycle. Figure 1-1 depicts the customer cycle, showing how a company converts the initial investment in marketing and sales to a completed, invoiced order. In developing the customer cycle for your company, you'll need to identify which groups are responsible for each of these stages. Make sure the relevant groups have basic agreement on "who does what" to double-check your model.

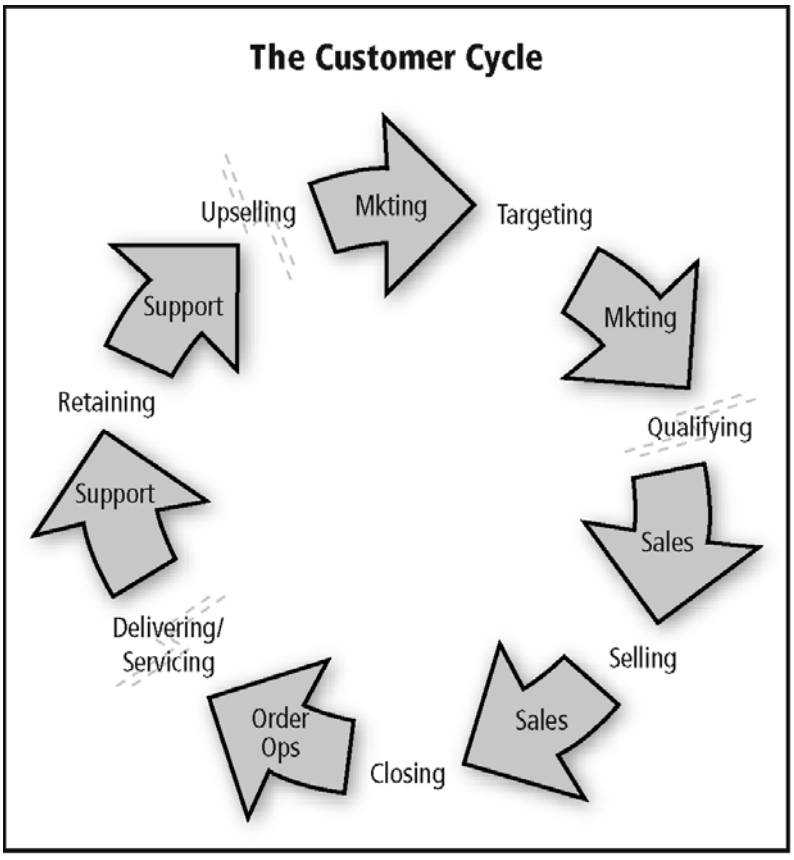

Figure 1-1
The Customer Cycle

Note the dashed lines at the qualifying, delivering/servicing, and upselling stages: these lines are meant to indicate where the current customer cycle "breaks"—where conversion or retention

ratios tend to be too low, or where organizations squabble and opportunities fall through the cracks. It is very common to have breakage points in these three stages, but every organization is different and you should identify the perceived breakage points for your organization as early as possible in the project.

The right-hand side of the customer cycle is the sales cycle, in which sales and marketing activities generate all your company's revenues. Because the sales cycle is the principal domain of an SFA/CRM system like SFDC, you'll need to have a detailed model of what the sales cycle stages are, who "owns" them, how long they take, and what the conversion ratios at each step are expected to be. Figure 1-2 illustrate an example sales cycle for a business-to-business (B2B) direct sales operation.

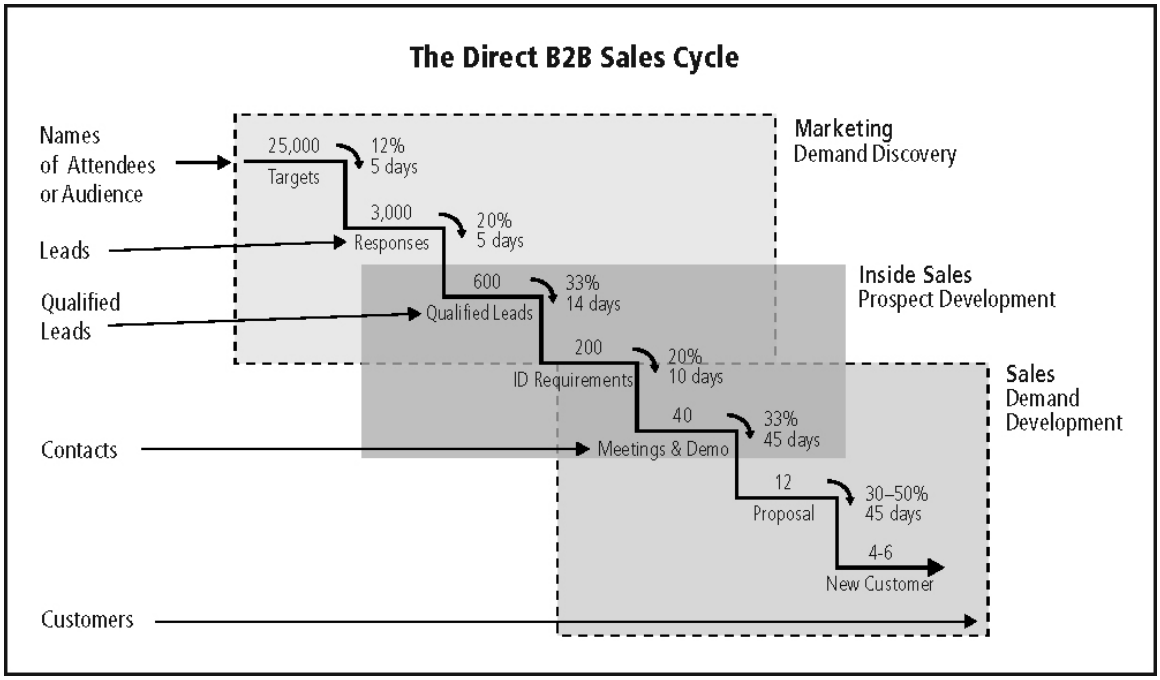

Figure 1-2
The Direct B2B Sales Cycle

The sales cycle may be viewed as a "waterfall" showing the conversion steps for each stage of the cycle. The diagram shows who owns each stage, what the expected conversion ratios are, and how much time each stage takes. Make sure that both sales and marketing executives concur with the basic data and relationships within this waterfall early in the project. Find out which stages represent the biggest problems and where the executives believe the SFA/CRM can help them the most.

Setting Business Goals

Before you drill down into any of the details, it's important to define a set of business goals that transcend individual requirements. These goals are best stated as numerical performance

objectives[12] that are not currently being achieved but are easily communicated to executives, such as these examples:

- Improving calls-per-day to 55
- Shortening the sales cycle by two weeks
- Increasing the conversion or close rate by 20%
- Lowering the cost of customer acquisition by 15%
- Increasing the average order size by $10,000
- Increasing revenues by 20%
- Increasing gross profitability by 10%
- Improving customer retention to 80%
- Increasing customer lifetime value by 10%
- Decreasing customer service complaints by 30%
- Decreasing incorrect orders to 1%

Of course, you can't have everything. Keep the list as short as politically feasible. But no matter how many metrics there are, upper management *must* prioritize them. These figures of merit will be important touchstones for prioritizing the features and process changes that will be done in the course of the SFDC implementation.

Setting Requirements: Who, Where, What, and Why

To make the business case for the SFDC project, you'll need to identify and prioritize requirements at an overview level. Even though the details of many individual requirements can't be known until several weeks or even months after the project go-ahead, you will need to have enough information to scope the cost and assess the benefits. You can't sell the project to management without identifying specific, concrete **pain points and potential advantages**.

The first step in requirements setting is to identify who will (and who won't) need to use the future SFDC system. It's easy to say "everybody," but that decision can mean big implementation, training, and ongoing costs. Worse, it means trying to make "everybody" happy, even if those people aren't critical users. In most companies, only a few departments directly touch the customer every day, so many people won't need direct access to the SFDC system. It's far better to have a narrow list of users, at least initially.

Create a spreadsheet with a list of user names, including the user's organization, job title, location, contact information, and expected SFDC interactions. From this list of users, group together the people who are expected to have similar SFDC interactions. Each of these types of user represents an *actor* (sometimes called a *persona*) in the system. For each of these archetypal actors, create a one-page description that provides overview information about them: location, educational level, job title, job responsibilities, preferences, habits, language, and so on. Almost any SFDC implementation will involve at least four actors, but this number will multiply quickly in large companies. To keep the number of actors to a reasonable level (more than 10 can be unmanageable), you'll probably need to generalize across similar actors. Descriptions of actors should cover their general *role and goals* in using the system: how often (and how long) they are

[12] Avoid vague or overly qualitative goals like "improve collaboration" or "gain visibility." This kind of goal statement is too mushy to drive a business decision.

expected to be on the system, what they are trying to achieve, which parts of the system they use, what they need to avoid, and what they expect in terms of usability (for example, users might need to do their job on a laptop while they're on the road).

Later in the project, team members will want to interview representative actors to get detailed requirements and feedback on the system. For now, however, these one-page summaries help provide some reality and context to the planning process. It's a good idea to give each actor a nickname and include a small photo on the description page to make the user seem more real. Log on to the *Salesforce.com Secrets of Success* book Web site (www.SFDC-secrets.com) to download a template Actor document.

Going beyond the list of people who will actually touch the system, it's important to create a list of people who will have an indirect dependence on the system. These individuals may never log in, but they will depend on some report or data flow that comes from the system. You don't really need to create a full persona description for these individuals (or roles), but it is important to understand which requirements they have for the system (e.g., a distribution center manager who might need the system to be fully operational on Saturdays, even if no users are logged in it).

Do You Need to Call in Outside Help?

This chapter assumes that the reader is part of a team that will be doing the project on their own. Many companies choose this path, but you should consider whether you need advisory help early on to prepare the business case, prioritize requirements, understand the implications of alternatives, and give guidance. The right advisor can save your company a bunch of time and money by avoiding blind alleys.

Consider an outside expert if you're feeling uncomfortable with the topics here or are looking for industry best practices from day one. An outsider can help you answer the following types of questions:

- What are your competitors doing with SFA and CRM? What's the customer expecting in this area?
- Which departments should be using the SFA/CRM system, and which ones really shouldn't?
- What are the high-payoff business processes to streamline, and which should be left alone (see Chapter 8 for more on this topic)?
- Should you try to enhance your existing SFA/CRM system, or should you do a wholesale replacement of it?
- What are the quantitative payoffs for your industry peers of a solid SFA/CRM system?
- What are the lessons learned and pitfalls to avoid in SFA/CRM projects for a company like yours?

Before you engage an outside consultant, check with the members of your own information technology (IT) team. They may have a list of preferred vendors and, at the very least, can help screen consultants for competence and value.

Get management involved early in the decisions regarding "how far do we want to go"—not the detailed requirements, but the general objective. Identify the executive champion, and engage

that person in the process (see Chapters 4 and 5 for more on this topic).Use the SFA Maturity Model™ Level descriptions from Chapter 5 to guide this decision, which will provide an overall "scoping" of the project investment. Also go through the SFA Maturity Model questions in Chapter 5 to understand your organizational readiness, and note which actors are part of particularly advanced or immature organizations. In your project planning, avoid over-reaching! It kills both budgets and the likelihood of project success.

Once the objective and the actors are settled, identify the *scenarios* (also called *use cases* or *user stories*) for each of the ways the actors will use SFDC. For example, actors in order operations will be entering and reviewing orders all day long, but they may also need to approve exceptional discounts or other special cases every day or so. Given that this approval process involves different thinking and actions from ordinary orders, approvals must be described in a different scenario. The same person may need to do sales forecast reports to support executives, again requiring a different scenario. These scenario write-ups should *briefly* describe what the actors are trying to achieve, the data they need to see and modify, the approvals they need to proceed, and the other actors or business processes they need to interact with. Make sure to note whether the scenario requires direct access to the system, or whether the situation could be handled through indirect means such as interacting with another application (e.g., the accounting system) or viewing a Web page, spreadsheet, or report. Log into www.SFDC-Secrets.com to see an example scenario.

From the scenarios, the team needs to distill a set of system requirements. In many cases, the scenarios will have overlapping needs that are stated from different perspectives. A key step is to identify and consolidate the near-duplicate requirements, and to map them to the features provided by SFDC and related systems. For example, one sales scenario is to create a quote for an opportunity. This is fairly similar to the sales operations' scenario of updating a quote as well as to the sales manager's scenario of reviewing and approving a quote. All these scenarios are the requirement to edit opportunity, forecast, and quote information in real time (with the proper access controls to protect privileged information). This requirement maps to stock features in SFDC's Enterprise edition, but adds the requirement to integrate quote approval with the accounting system. It's a smart idea to create a correspondence table showing which scenarios need each requirement. Check out www.SFDC-secrets.com for an example of this kind of mapping.

Many users (particularly executives) don't really think in terms of discrete requirements. Instead, they simply visualize what they need to make a decision or take an action. For this reason, it's a good idea to do mock-ups of a few reports, forms, approval screens, and dashboards (using Excel, PowerPoint, or even pen and paper) and do a "day in the life" walk-through of people's jobs to get them thinking. Creating these mock-ups can very rapidly elucidate hidden—and difficult—requirements. You'll be surprised how often a user will say, "Of course, I have to be able to see the customer's order history to make my decision . . ." To make sure you're being realistic, run through a couple of the exercises described in Chapter 2.[13]

Watch out for internal contradictions

Sometimes the requirements of different teams will slightly (or even thoroughly) contradict the stated goals of other departments. This is always dangerous to system credibility, so it's important to flag these contradictions right away.

There's no way for the SFDC team to resolve mis-alignments or contradictions among goals: that has to be done by a meeting of the

[13] SFDC will have no problem complying with all these requirements, but you need to know about them so the system can be properly configured from day one.

minds at an executive level. One side or the other will have to subordinate their objectives for the good of the whole.

Resist the temptation to speak in terms of product feature-lists. Each requirement should be stated in terms of the business need (i.e., a step in a business process with a measurable result) rather than as a technical description (how the feature works). Each requirement description should be as brief as possible, and each should be stored in a separate document. (If your company is really into paper, the requirements document should be placed in a three-ring binder, with individual pages being inserted or updated on at least a monthly basis.)

In addition to user-driven features, make sure to cover "taken for granted" system characteristics in your requirements documents: system availability (e.g., 7 A.M.–10 P.M. EST five days a week), system response times (1 second during operating hours), administrative response times (e.g., 1 hour during operating hours, 4 hours on weekends), thresholds for data error (e.g., zero defects in records A, B, and C, but 4% in all other records), international language support, multi-currency support, unusual fiscal periods, and other "environmental items" that you discover from your requirements gathering.

In a large company with complex business processes, the scenarios and requirements will multiply quickly: their sheer number will soon be overwhelming. The best way to handle this issue is to create a short overview spreadsheet that organizes and summaries the requirements in a hierarchical fashion and helps the team visualize them from a "top-down" perspective. This high-level spreadsheet should not provide the details of any one requirement, but rather should show how the requirements fit together to form the big picture of the project. (Check out www.SFDC-secrets.com for an example requirements overview spreadsheet.) The requirements spreadsheet needs to make it obvious to team members (and executives) the reasons why each line item is there, its scope, its schedule, its cost, its sponsor, and the line item(s) it could be traded off against.

Organizing and Publishing Project Documents

Even before the project begins, best practices call for storing, publishing, and organizing the project's documents online. Put them in an intuitive hierarchy, with all the project-related documents in one place that is scanned by an internal search engine or a desktop index (such as Google's free desktop search or X1's indexer) and is backed up regularly.

You can store these documents in a shared folder on a file server, but even better is to have them in an intranet site. Best of all would be a wiki[14] that's accessible to all team members (including contractors). Your top-level wiki "article" should include pointers to the project schedule, the current phase's goals and nongoals, project personnel, news, discussion forums, blogs, and user tips on the system. The wiki should also provide a hierarchical library that includes all of the requirements documents, discussions, rationales for priority calls, and the project requirements phasing spreadsheet.

[14] What's a wiki? Time to catch up on your jargon! A wiki helps you present information in a *kiosk model*—where users are given information only when they care about it. This approach stands in contrast to a broadcast model like email—where users are given information at a time when the *originator* thinks it's relevant, rather than the receiver. Using a wiki properly can save dozens of emails, phone calls, and even the occasional status review meetings. To find out more, read the article on the topic of wikis at the grand-daddy of them all, wikipedia.org.

The wiki should be "Information Central" for the project, so any relevant information should be available there, including team members' contact information, email aliases, email/discussion archives, cheat sheets, podcasts, WebEx sessions, screen cams, and internal success stories. By making it really easy to get to and find all the current project information, you'll be on the road to clearer communications, better expectations management, and a much lower chance of confusion throughout the project.

You should also put a short document into SFDC's Documents section describing how to get to the file server or wiki. Promoting what's there will allow any interested user or project team member to quickly become better informed.

It's usually best that these documents be editable Word files, with change tracking turned on. For Excel files, the first page of the spreadsheet should include a revision history indicating who changed what and when. The files should be downloadable by any authorized user, but should be *updatable* only by sending the changes through the project manager. In this way, the project manager can filter and prioritize input and manage the evolution of the requirements and other documents (which inevitably change throughout the project).

Documents should never be deleted. Obsolete documents should be marked as such and moved to an archive directory.

For certain kinds of files, it's useful to leverage the shared editing and shared access of Google Docs. Although the files have to be much simpler (no macros or heavy formatting), Google doc sharing can improve collaboration in a distributed implementation team. This is particularly true for status documents, priority lists, or other highly collaborative content. SFDC has recently created an integration with Google docs that makes this option even more tempting.

Prioritizing Requirements

One of the toughest tasks throughout the life of the project will be understanding the ramifications of requirements and making priority calls among them. With a system of any size, some requirements will have to be delayed or may even have to be abandoned as too expensive. Making these decisions can get interesting when interdependencies among the requirements exist, and things get even juicier when political overtones and competition among the requirement sponsors complicate matters.

Although this complexity sounds scary, there's a counterbalance. The overriding principle of prioritization for any SFA/CRM project is that **it doesn't matter how many requirements you've delivered on** if the users aren't adopting the system in droves. This is because the *value* of an SFA/CRM system—to sales, marketing, support, and executives—grows in proportion to the amount and quality of data the system holds. Make priority calls that cause users to get quality data in the system sooner and that persuade users to get on the system more often . . . and the rest will follow. Instill this overriding principle in the project sponsors as much as you can so everyone pulls in the right direction.

One way to guide prioritization discussions is to use the short list of the specific business problems that the executives defined earlier in this chapter. These specific improvements should be specific, unambiguous, and quantified—"reduce customer support response times by 20%," not "improve brand value." Estimate how the major features will affect these business goals, and from that calculate the return on investment (ROI) for the feature: lowering of costs, increasing revenues, or both. This overview page will help keep things in perspective as you try to make priority calls.

Like all complex decisions, the tough calls will be made in meetings where there is insufficient information. Your requirements summary spreadsheet is the tool the team will use to make those tradeoffs and priority calls. The goal of the project leader is to channel the discussions during these meetings down paths that are rational: all of the choices are achievable, and the final choice optimizes ROI.

> **Prioritization Tools**
>
> Appendix A describes several tools that can be used to prioritize requirements. Give at least one of them a try to find out which tool is most natural for your team and produces the most credible priority rankings.

The two groups that usually get the highest weight in the prioritization are sales representatives and executives. Things that are of direct, immediate benefit to them are assigned a lot of points. The specific points you give to other departments and user types are up to you, but an SFA implementation team forgets "who's the boss" at their peril.

There isn't a universal formula for prioritizing requirements: too much depends on the specifics of your company. For example, should the requirements from highly sophisticated users (as identified in Chapter 5's SFA Maturity Model) be prioritized *higher* than others (because those users can gain the most) or *lower* than others (because those users will be better able to cope with an incomplete feature set)? The answer to this question inevitably depends on your company and its management style.

You'll want to carefully plan out who will start using the system when, as the timing affects priorities. Put simply, the timing and order in which you deliver SFDC functionality and extensions can have a *big* impact on the total project cost and schedule. Some functional areas go together very easily; others can involve a lot of controversy, meetings, and rework. While the order of functions is fundamentally your choice, a good rule of thumb is to bring users onto SFDC in the order listed in Table 1-1.

Table 1-1
General Sequence of Bringing Users into SFDC

Organization	Activity	Business Priority	Political Priority	Phase
Telesales/inside sales	Orders	High	Medium	I
Order operations	Approvals	High	Medium	I
Sales Development	Appointments	High	Low	I
Telemarketing	Cold calls	High	Low	I
Marketing	Lead generation	Medium	Low	II
Sales analyst	Executive support	Medium	Medium	II
Product marketing	Purchase analysis	Medium	Low	II
Sales VP and Marketing VP	Management, forecasting	High	High	II
Executive team	Management	High	High	III
Legal	Contracts High	Low	III	
Sales engineers, consultants	Deal support	High	Low	IV
Shipping/receiving	Distribution	Low	Low	IV
Customer support	Customer care	Medium	Low	IV

Finance	Business analysis	Medium	Low	IV
Field sales managers	Sales management	High	High	V
Partner/channel manager	Channel management	Medium	Medium	V
Individual sales reps	Selling	High	High	VI
International operations	Selling	High	High	VII

Almost always, the revenue-focused business processes conducted at headquarters are implemented first. Later, other headquarters processes are brought online. Usually, business processes done in the field happen a bit later.

You might wonder why the executive suite and other high-priority roles in the company seem to come so late to the system. Why aren't their requirements satisfied first? Chapters 3 and 4 answer that question: the quality, completeness, and relevance of the data in the system just aren't good enough early on. Any reports or dashboards an executive wants won't be reliable, and they could even provide misleading information that would lower the credibility of the system. It's a big mistake to bet the farm on SFDC data assets before they are ripe.

Add groups and adjust priorities on the list in Table 1-1 to fit the realities of your business. No matter what the order selected, use the ordering to help prioritize requirements. Requirements coming from teams that will be coming on board early should be treated as having a higher priority (so they get done early).

A key issue in prioritizing requirements is the "squeaky wheel" phenomenon, where a noisy or politically powerful proponent causes great emphasis to be placed on a requirement that isn't really critical. All too often, a significant proportion of the effort in large projects is devoted to requirements that are "nice to haves" but that ultimately waste time and effort. Consequently *the* highest leverage thing a project leader or business analyst can do is identify and deprioritize the uncritical. While avoiding confrontation, smoke out dubious requirements using gambits like these:

- "How much of *your* budget would you be willing to dedicate to solving the problem?"
- "Can you quantify how much waste this problem has been causing for you on a monthly basis?"
- "How much profit increase would the company see if this were done?"
- "How much more productive would you be if the problem were solved?"
- "Which other department needs this feature?"
- "How have you been able to succeed without this so far?"
- "What's the forcing function—the deadline beyond which we can no longer do business the current way?"
- "Which of your other requirements would you be willing to put on the back burner to get this one done?"

Avoiding Happy Ears

Whenever a new system is being developed and people are interviewed about what they need, users start to hear hints about how things will

work. Nearly everyone will assume that if they've heard about an issue, it's going to be solved by the new system. They'll even extrapolate, imagining all the wonderful new features that will become available to make their job easier. It's human nature to be optimistic, and that goes double for sales and marketing folks. But this "happy ears" phenomenon leads to spiraling expectations and scope creep that can kill a project.

Project leaders and especially project sponsors must continuously push to lower these expectations, even if they yet haven't seen overt signs of happy ears. The project goals must be simple—even minimalist—for the first phases, and "great ideas" must be explicitly pushed off in time. Even when things are going okay, *undersell* and be tentative about making the schedule. Even if the team has a way of delivering "something great," consider delaying it (ironically, see "The Art of the Quick Win" later in this chapter). When you *do* deliver this great thing, *do not publicize it until it's delivered,* so that you can provide pleasant surprises to users.

In prioritizing and trading off requirements, the project leader will need a set of executive sponsors who can act as champions for political and resource issues. For SFDC project success, the champions *must* include the VP of Sales, but usually include the VP of Marketing, the CFO, or perhaps even the CEO. These key supporters cannot afford the time needed to participate in the project, so they'll need to deputize a senior person on their staff to be a regular participant[15] on the SFDC implementation team. The project leader should form a management council of these deputies to make priority calls, policy decisions, and tradeoffs. For simplicity of voting, the team needs to be small and contain an odd number of people (including the product manager). A small management council would include representative from the Sales and Marketing departments, plus the project leader. A larger council might add representatives from the Finance, Customer Care, International Operations, Professional Services, Channel Operations, and Business Analytics departments. This council will need to have *brief* meetings on a weekly basis, and it should devote a significant amount of time to analyzing business processes. Because a key factor in ensuring project success will be the availability of the right people for this team, get their (and their bosses') commitments early.

The challenge for the project leader is to keep one (and only one) priority list that encompasses everyone's needs, while clearly limiting the number of items that are "must do's" for each phase. Even so, maintaining a consistent, clear, tightly enforced requirements priority list is an invaluable tool for the project leader, and its existence will help fend off countless arguments during the project.

When Requirements Should Bend

It is common for requirements to be stated as absolutes, with intricate detail being provided about the way things must be done. But these "requirements" are often an interpretation of a business need, or an executive's preference, or even a legal regulation. Sometimes, the literal requirement

[15] As we'll discuss later in this chapter, "participant" means an active worker who will be charged with the task of delivering documents and making binding decisions. Participants' effort on the team needs to be part of their "day job," so their boss needs to dedicate a portion of their schedule and allocate some of their personal goals (MBOs) during the months of their project involvement. Their effort needs to be measured and rewarded!

is a poor interpretation of the underlying business need. It's important to be as creative as possible in requirements statements so that you don't over-specify or get locked into one particular approach. Identify alternatives and different ways of achieving the underlying goal.

For example, the Finance department may have an edict that an order cannot be shipped without a manual approval. So the requirement is stated as a mandatory human approval cycle. But if the approval cycle is there only to apply a set of strict rules, maybe the *manual* approval cycle isn't the real requirement. By restating the requirement as "no shipment will be made without applying the following rules," it becomes possible to have a fully automated approval, saving time and cost on every order.

Look for opportunities to restate "requirements" that are over-specified or arbitrarily complex. Requirements should be statements of business goals and criteria, rather than strict step-by-step procedures.

In some cases, the best path forward is to change the business process, rather than investing in automation of a silly or obsolete practice. In particular, look for opportunities to make things better by making subtle changes to the sales and marketing processes. Some of the most doctrinaire SFA/CRM requirements miss key opportunities for leverage. The SFA/CRM system will give sales reps and managers information and tools they never had before. Because the whole point of the system is to make it easier for the reps to make their numbers, why not take the blinders off? For example, better qualification of leads can mean fewer pointless sales calls and shorter sales cycles. Likewise, marketing personnel should be encouraged to think outside the box, looking for new ways to plan campaigns and execute events. Of course, you'll need to get the executives' approval before you formally recommend process changes or restate the requirement, but it's well worth the effort.

Bending and restating requirements in this way can make the system easier to implement, easier to use, and more beneficial to the business over time.

Knowing Your Boundaries

One of the most important issues in requirements setting is determining the "edges" of the system—that is, the boundaries beyond which users must log in to a different system if they want to do something. All users would like the system to encompass all of the things they need to access for their particular jobs, but it's impractical to deliver a single system that covers every business function for every employee.

When you first install SFDC, it has the following functional boundaries:

- **Order entry:** Anything having to do with bookings, invoices, or payments is in your accounting or Enterprise Resource Planning (ERP) system, not SFDC.
- **Order management:** Anything having to do with shipments, order status, returns, or exchanges will be in the distribution or ERP system, not SFDC.
- **Marketing:** Leads, contacts, campaigns, and simple email blasts are fine within SFDC. Almost all of the details having to do with emails, marketing events, advertising, or other marketing initiatives, however, will be in your email blasting, marketing automation, or content management system, not SFDC. For example, SFDC tracks the outcomes and history of campaigns, but actual campaign management is external to the system.
- **Customer assets, inventory, and licenses:** Almost anything having to do with

the customer's order history, shipments, downloads, or licenses will be in your accounting, ERP, or other corporate databases. SFDC has a very limited view of assets.

- **Defect reports or bug tracking:** Anything having to do with the technical side of customer problems will be in the defect management or bug tracking database; most companies already have an entrenched system for this purpose. SFDC can be configured to integrate with these systems, but out of the box it's really focused on "cases" or "incidents."
- **Documents:** Almost anything in your sales literature, marketing library, Web site content, proposals, customer specifications, or contracts is already stored outside of SFDC, often in a file system or in content management products. SFDC has an add-on product that helps present and organize document libraries in a clever way, but this product will usually be a supplement to existing external systems, rather than a replacement for them.
- **Business analytics:** Although SFDC provides an excellent set of built-in reports and dashboards, almost any business analyst will need to go beyond these features and use an external database and business analysis tool.

For a detailed view of business systems that touch on SFDC, see Figure 1-3. As shown in the diagram, several add-on products may be tightly integrated with SFDC to improve its functional coverage (see Chapter 7 for a discussion of these products). These extensions and external products are a huge strength of SFDC, because they allow the core system to be simple and easy to use, yet flexible and scalable enough to handle much broader requirements. Further, SFDC has a very open and well-documented set of external interfaces, presented as Web services. These "SOAP APIs" allow programmers to connect, integrate, and extend SFDC to almost anything.

Even though you *could* integrate SFDC with any external system, it's easy to overshoot the mark by trying to do too much in the initial implementation. In evaluating and prioritizing requirements, it's important to identify when a requirement crosses over into an external system. Any such requirement is inherently more complex and risky, so it should be assigned a lower priority than requirements that are entirely within SFDC's native functionality. Read the section on integration in Chapter 7 before you deal with any requirements for integration.

Thanks to the modularity of the SFDC architecture, almost all internal functions can be turned on in phases at the time of your choosing. For example, turning on the Campaigns or Forecasting function can be done the day you bring up the system or, alternatively, months later without involving significant rework. Many external add-ons act in the same way, allowing a very flexible deployment schedule. However, some of the more sophisticated add-ons (such as Eloqua marketing automation or Intacct accounting) will involve significant changes to your data, the user interface, and user behaviors. Consequently, the timing and sequence of deploying the more complex and expensive add-ons cannot be taken lightly.

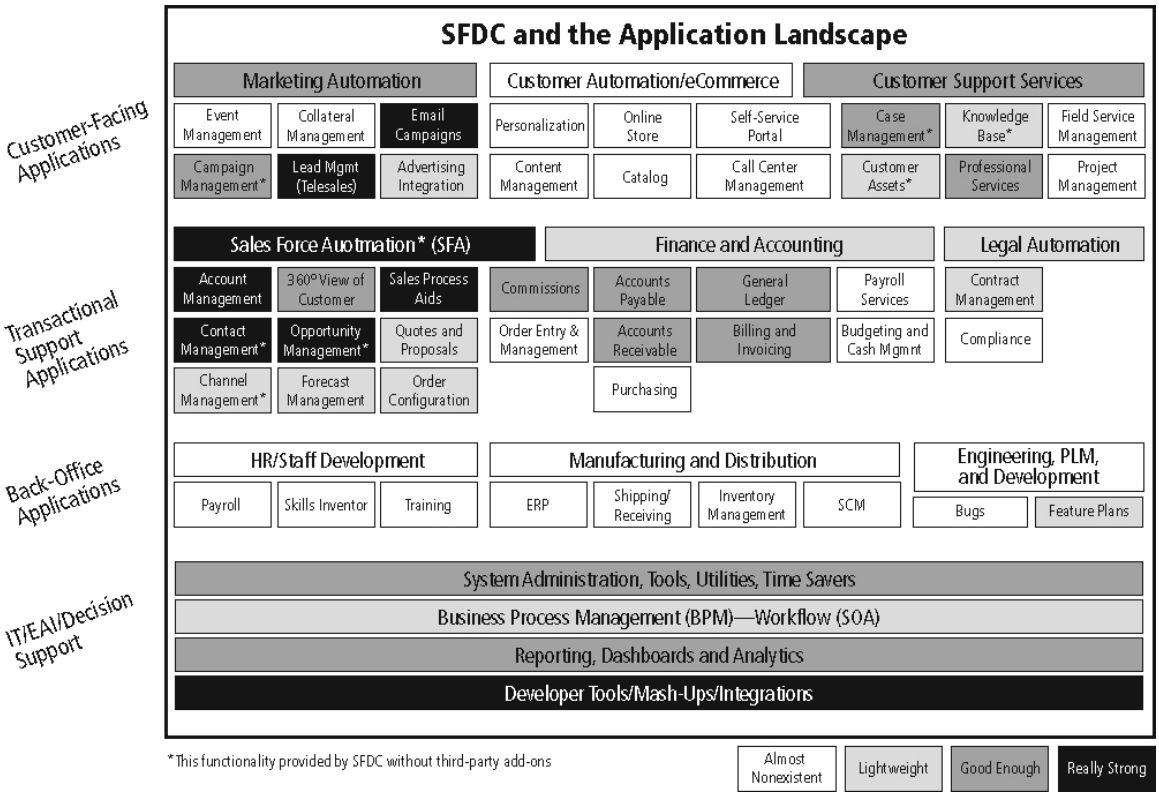

Figure 1-3
SFDC and the Application Landscape

When Other Problems Must Be Solved First

In large organizations, the political drivers for installing or replacing the existing SFA/CRM system will be executive visibility, customer reporting, or uneven execution of business processes. Unfortunately, merely upgrading to SFDC will not help in many of these situations. In large organizations, other problems need to be solved *before* the new system will function properly, let alone resolve any deep issues.

The project manager should analyze the requirements and problem statements to understand which of them are actually caused by bad data, disconnected systems, unclear business rules, inconsistent business processes, and other "environmental" or infrastructure issues. These problems are typically scattered across several databases or applications, and fixing them will be a prerequisite to any successful SFDC deployment.

If you discover this situation, it is important to get a project team started on analyzing and remedying those external problems even before your SFDC work starts. That project team should be chartered to work with

your SFDC team, but typically it should be managed separately. If this team is chartered to work under the SFDC team, make sure that its focus doesn't become too broad (e.g., "rework this entire outside system") or distract the main SFDC implementation team.

As with all requirements, the extension of SFDC via add-ons and integration with other systems should be prioritized with a keen eye toward schedule and risk. Even though basic integration is of the "plug and play" variety, the interaction among systems and databases can cause subtle data corruption problems that won't discovered for several weeks. Consequently, it is usually a good idea to deploy basic SFDC functionality without external integration, get the users comfortable using it, and get some real data flowing through the system (even if only manually) in the early weeks. Later, new functional add-ons and integrations can be deployed on top of a stable baseline system, and users can gradually become used to the new features as the system expands. External integrations (particularly with complex systems such as accounting, ERP, and marketing automation) should be started early for testing purposes, but not turned on for full two-way data transfer until later phases.

Appendix B provides example requirements for both a small company and a large company. Of course, every company's needs and priorities will be different, but the examples show the level of detail that you will need (not much at this stage) and the kind of interdepartmental prioritization you'll need to think about when preparing your company's list of requirements.

The SFDC system will almost certainly be implemented over several quarters, with only the first two phases as "hard priorities." Later requirements will be reordered depending on perceived business priority after initial usage; technical issues or needed process changes will be discovered during the project. In addition, perceived business needs may change over the life of the project. It's important that the executive sponsors understand, however, that each major requirements change will cause some delay and wasted effort. Communicating this point clearly but pleasantly is one of the key challenges for the project leader, because the executives are as vulnerable to "happy ears" as anyone else.

Making the Business Case

Most companies will require a formal business case for an investment in SFDC. Even if yours doesn't have a formal review meeting for the decision, it's a good idea to go through the exercise of cost, benefit, and ROI analysis so that the team understands the financial implications of what they are doing. Understanding the things that really make a big financial difference—and the things that really don't—helps everyone make better decisions throughout the life of the project.

Estimating Costs

When evaluating Software as a Service (SaaS) systems such as Salesforce.com, it is tempting to think that cost analysis isn't critical. After all, it's just a monthly fee and you pay only as long as you're receiving value . . . right? That's what the salesperson wants you to think, if only to make you stop thinking. Life isn't that simple.

Procurement

The big three cost areas are procurement, implementation, and ongoing user expenses. Procurement costs for SFDC, like most SaaS products, consist of a monthly fee. The fee varies based on the following factors:

- The number of users (internal "full feature" users, platform users, partner users, and customer users)
- The version you buy (most readers of this book will purchase Enterprise edition, but many will also want the Mobile edition, the Sandbox, or other extra-cost items)
- The length of the contractual commitment (one year is the minimum; more than three years seems to make little difference to the discount)
- Your willingness to prepay (prepaying one year makes a difference, but longer prepays seem to make less of a difference)

Who Should Pay?

The best decisions will happen when the departments that *benefit* the most *pay* the most for the system. This typically means sales should pay at least 50% of the total, with marketing and support sharing the remainder of the cost. Of course, this split depends entirely on your company's organizational structure and system usage.

Another funding formula is to have the project be financially sponsored by the executive suite—at the corporate or business unit level, which shows executives' level of commitment.

The only funding model that *doesn't* work is to have central IT foot the bill. While this works with infrastructure, in SaaS applications—and particularly those that directly benefit only a part of the company—this model doesn't work in either the short run or the long run. Check out Chapter 6 for more on this topic.

Because you're gong to be delivering the system in phases, make sure to activate only the number of seats you'll actually use at any one time. In the negotiations for a large deal, this aspect of incremental deployment may save your company a significant sum.

Include add-on products in your procurement cost estimates, thinking through phasing effects that will affect the number of users and timing of activation for each product separately. Most add-on products for SFDC are also SaaS, but a few are on-premises products with hardware requirements and perpetual licenses. For those products, don't forget to include the cost of hardware procurement.[16]

Do not neglect support fees. The basic support level that's included in SaaS products will be insufficient for at least your first year. You'll need speed of access to in-depth expertise that is available only through premium support. Premium support is really good, but pretty expensive: make sure to negotiate for premium support only for the number of users you actually expect to

[16] For both security and performance reasons, you'll almost certainly want any on-premises add-on products to run on dedicated servers.

have during the initial deployment phases. Your team members (particularly developers and system administrators) will also need technical training classes (typically $2500 each) to become rapidly productive. Do *not* negotiate for these items piecemeal if you want the best discount.

If your company will have more than 100 users, the annual fees for SaaS can be a surprise. While it's true that you aren't paying for upgrades, patches, hardware, and operating staff, SaaS fees can easily exceed $100,000 per year, and the amount climbs to more than $1 million per year for the very largest deployments.[17] Don't assume that you'll have a lot of bargaining power once the deal is signed: the only way you can get a better discount is to buy even more or to commit for an even longer period. Downgrades are not allowed. The vendor knows you don't have a credible threat of changing the system, as the real cost of a changeover can be quite large even with the flexibility provided by SaaS's Web services technology. As your users grow accustomed to features and your developers become experts at using SFDC's APIs, you will be just as locked in to the vendor's technology as you would be with any Microsoft product.

To get a significant discount on the procurement, you'll need to get a budgetary allocation for the multi-year commitment. Work with your Finance department to develop the business case, making sure those personnel understand that the number of users will go up over time even if the per-seat price is stable.

Implementation

In contrast to "procurement costs," implementation costs for SFDC and other SaaS products are one-time fees. You will almost certainly need consulting and other services required to convert legacy data, integrate with other systems, write custom `APEX` and `Visualforce` software, test the system, deploy it, and train users. Do not assume that your internal IT people are interested in SaaS projects or that they have the required skills and experience (but check out Chapter 13 anyway). Some of the implementation costs are easy to identify and are "fixed fee" in nature (e.g., training classes). Most of the big elements of implementation, however, are highly customized and can be brought to a fixed-price bid only after significant analysis by your vendor. Of course, every Finance department will want a fixed price, but there is *significant* gamesmanship in the bidding process. It is very easy to get caught up in the engineering change order (ECO) trap, where "fixed price" contracts create an avalanche of over-runs.

When the Price Isn't Right

At the beginning of a project, all you can see is price. At the end of the project, all you'll remember is the value of benefits, the quality of the work (or lack of it), the pain (or lack of it), and the over-runs (large or small). It's easy to be "penny wise and pound foolish." **It is far more important to choose a vendor that you can trust, rather than one that bids low.** You need to see proven experience, skills in project management, actionable communication, and references from other customers.

One of the advantages of the phased deployments we recommend in this book is that the vendors can give you a more accurate bid for the

[17] Discounting on SaaS deals is much less than it is for traditional enterprise software. Discount levels for the very largest deployments (10,000 users or more) are a well-guarded secret, but it is unlikely that discounting exceeds 80% even at that level. If you have knowledge about the negotiated price of a large SFDC deal, feel free to email me at david@SFDC-secrets.com.

immediate phase (which will be very tightly defined) than they can for the project overall (which may suffer from scope creep and extensions). *Best practices are to use hourly rates with a tightly managed cap rather than fixed-price engagements* because this practice (1) exposes the scope creep that hurts overall chances of project success, (2) avoids the bickering, administrivia, and gamesmanship of fixed-price "change orders," and (3) prevents vendors from hiding large margins in their bid.

Explore your consultant's willingness to work using performance incentives. If they are designed properly, these types of contracts can yield the best possible deal for you because your goals and metrics are tightly aligned with the consultant's. As wondrous as pay-for-performance contracts are, they're equally rare. But give it a shot.

The first step in quantifying the implementation costs is drawing up the statement of work and development/integration requirements for each of the projects. The more homework you can do yourself, the more accurate the vendors' bids will be. Some of the project areas will require analysis that you can't do, and the bidding vendors will have to analyze this themselves. The vendors are unlikely to do enough "drill-down" analysis for free, but there's good news: this scoping project will be short and *should* be fixed-price. This is money well spent before you make a large commitment. Some vendors will do this scoping project only *after* you have signed on with them for the main project. My own experience with this kind of vendor is not good, but don't treat that issue as a show-stopper. If the vendor is very well qualified and you trust the consultant, proceed.

Note that the cost of implementation talent for serious SaaS projects isn't that much different from the corresponding costs of conventional on-premises software packages. Skills and deep experience in SaaS implementations are fairly rare (particularly outside the United States), and fees can be $300/hour or more for the best people. Many customers have experienced overall implementation costs that exceed the software fees they pay to their SaaS vendors in the first year. Although this amount is significantly fewer dollars than customers paid for classic enterprise software (where implementation was typically projected to be 1.5 or more times the cost of the software licenses), it may still come as an unpleasant surprise if proper allocation for it has not been arranged.

Do your cost estimates only for the initial implementation, but know that the happier your users are, the more likely they will be to ask management for further system enhancements. It is not uncommon for companies to have multiyear engagements before they reach the management nirvana of the 360-degree view of the customer.[18]

Don't forget the costs of your internal people. It's not unusual to have five people be involved on a full-time basis in an SFDC implementation (even with consultants), with another five involved intermittently. Even though these people's salaries are already budgeted for, the effort they put into SFDC will be time that won't be available for other company priorities.

[18] This refers to the ability to see all customer interactions with your company from all angles, from Web site visits to payment history to letters of complaint. This visibility lets management see which accounts are the most valuable versus which customers are noisy but just not profitable enough. While this 360-degree visibility is very powerful for marketing and management personnel, it involves extensive integration and changes to business processes that can take years to come to fruition.

Ongoing Costs

After estimating total first-year costs—first-year fees, add-on procurements, and implementation expenses—the ongoing costs for SaaS basically consist of the annual subscription fees. There may be some one-time costs in the out-years (such as a data recovery exercise or temporary use of SFDC's Sandbox), so put a placeholder in those budgets. The more interesting ongoing costs are related to people: follow-on implementation or expansion work, training costs for new users or administrators, travel and fees for power users attending technical conferences (a good investment), and time for any consultants working on the system.

As with any large system, data pollution is an inevitability and is poisonous to the system's credibility. Business analysts and others will discover corrupted or duplicate data creeping into the system over time. Whether it's a temporary coder, a data-entry clerk, or overtime hours of internal people, some corner of the budget should be set aside for a health check and cleanup session at least once a year. You'll almost certainly want a deduping or data cleansing tool, which will cost at least $3,000 per year, which is money well spent (see Chapter 7 for discussion of this). Furthermore, identifying the source of the problem, rectifying it, and cleaning up the data are often tasks carried out in collaboration with contractors.

These ongoing costs are likely to be most pronounced in year two of the system, but some budget should be set aside every year after the initial deployment.

Quantifying the Return

Any project this size has to be justified with improvements to the bottom line. The two components of achieving this goal—lowering costs and increasing revenues—are hard to quantify. But they can at least be modeled to provide a credible SWAG.

Lowered Costs

To counterbalance all of the costs outlined previously, it's important to identify areas of cost saving. Because SFDC is a salesforce *automation* system, you'd expect to find some. But that's rarely the way it works—after all, many of the expense areas are sunk costs that cannot be recovered or labor that is taken for granted. You may have to be a good detective to ferret out the labor savings.

The first obvious place to look is cost avoidance for the system SFDC is replacing. It won't be much, but at least it can be immediately quantified. Next, look at ancillary systems that can be decommissioned because SFDC makes them less relevant. For example, you may have a large number of spreadsheets for the marketing group with macros that won't need to be maintained or extended any more.

The next place to look is wasted labor. Whether the waste takes the form of contractors or employees, the hourly rate should be quantified. The first waste-avoidance item will be system administration and maintenance for software and hardware that can be retired. This will probably amount to at least 32 hours per month; for large systems, it can be much more. Another source of wasted labor is sales administrators who—even in small companies—may spend 8 or more hours per week preparing the weekly forecast, and order operations people who spend 20 hours or more in order entry, correction, and reconciliation. Some business analysts also have to spend huge chunks of time trying to piece together spreadsheets from fragments of customer data. In addition, order fulfillment people (license generation, shipping, distribution, and expediting) may spend

their entire day doing things that could be 80% automated. Finally, customer support people can easily waste 10 minutes per call trying to find data, correct erroneous entries, or fix problems that could have been avoided by automation. These areas of direct waste add up, as they occur every week and may become much larger in the closing weeks of the quarter. Even though employees' time is a "sunk cost," quantify the savings as if those workers were paid by the hour.

See Table 1-2 for an example cost-savings matrix for a small company (as always, visit the book's Web site to find free downloads of Excel and other files that you can use on your project). Check out www.SFDC-secrets.com for other examples.

TABLE 1-2 Potential Cost Savings and Efficiency Improvements

Cost Savings	Units	Savings/Year
Hard Cost Cavings		
Ongoing license fees for existing SFA/CRM system	n/a	n/a
Support fees for existing SFA/CRM system	n/a	$2,000
Hardware maintenance and support costs for SFA/CRM system	1 server	$1,500
Auxiliary software products for existing SFA/CRM system	3 licenses	$3,000
Potential Cost Savings/Efficiency Improvements		
Tradeshow and related travel expenses reduced	1 tradeshow/year	$30,000
Advertising and other marketing expenses reduced	n/a	$20,000
Avoiding unproductive on-site sales calls/demonstrations	1 meeting/year	$2,000
Avoiding low-probability "proof of concept" projects	1 POC/year	$2,000
Improved product marketing and engineering decisions	n/a	$10,000
Improved executive decisions	1 decision/year	$25,000
Internal Labor Savings		
System administration time for existing SFA/CRM software and hardware	16 hours/month	$20,000
Sales administrator time reconciling orders	32 hours/month	$30,000
Sales administrator time rolling up forecast	32 hours/month	$30,000
VP time investigating/adjusting weird forecast numbers	12 hours/month	$12,000
Missed Opportunities		
Lower cost of customer acquisition by 5%	50 new customers/year	$100,000
Prevent one lost sale*	1 new customer/year	$50,000
Ability to do one more upsell*	1 upsell/year	$20,000

* This should be the revenue from the average first sale to new customers. Sophisticated finance types will tell you that this is an overstatement, and will try to get you to use the *profit* from a new customer instead. Don't do it: your fall-back position should be the **contribution margin** value from an extra transaction.

Opportunity Costs

The larger cost savings are more difficult to quantify because they involve missed opportunities or avoidance of wrong decisions. What would it mean if the cost of customer acquisition were 5% lower? How would it lower costs if the sales cycle were a week shorter? What would be the impact of reducing sales representative turnover by just *one rep* per year? What percentage of marketing dollars is spent on marketing to the wrong people or participating in the wrong events? How much could be saved if you could quantify which marketing activities really produced revenue, rather than just "visibility"? How much travel expense and labor could be saved by avoiding just one unproductive tradeshow, sales call, or proof-of-concept project? If you knew more about which prospects were (or weren't) going to yield the most profit, which other improvements could your company make?

Focus on quantifying **external waste** that could be reduced, rather than on internal jobs that could be eliminated, as the latter will bring the project nothing but political strife. You'll need to interview people in sales, marketing, and pre-sales support to make this determination. Nevertheless, as you develop an effective model for estimating the cost savings, the amount saved can turn into big money.

There is also value to the risk reduction that SFDC will bring. Hits to customer reputation, knocks to brand value, and even Wall Street embarrassment all cost the company money—and all can be improved by a really solid SFA system. Avoiding just one unreliable forecast or a few irate customers can save enough to pay for the whole of SFDC. The problem, of course, is quantifying this value.

> **Head's Up!**
>
> Watch out for finance groups that try to use your estimates as a justification for cutting the budget of other departments. They may say, "Well, if you do such a great job detecting unproductive marketing activities, we'll just cut their budget by 10% now to make sure you actually achieve that goal." This kind of budgetary reasoning can get poisonous in a hurry, so always couch cost savings as *potential* and as a way of increasing the leverage of the money that will be spent anyway. In other words, describe the change in terms of improved ROI or getting more value for the company's money, *not* in terms of spending less money.

Increased Revenues

The flip side of lowering costs is increasing revenue. To develop credible estimates, it's important to interview several sales and channel managers, discounting the opinions of people who are not directly responsible for revenues. It is political suicide to present your revenue conclusions without having first reviewed them with the Sales VP.

Develop a spreadsheet that shows the value of extra deals that could be closed with the aid of better prospect and customer information, better lead quality, tighter qualification, and higher conversion/win rates. What if a hot prospect never "fell through the cracks"? What would be the value of the extra deals you could do if the sales cycle were shorter? What kind of extra revenue might the channel produce if leads were handed off and managed more effectively? What would be the value of doing one additional "upsell" per year or of closing a deal with one more customer? What if your customer loyalty or retention rate increased by 10%? For the sake of credibility, keep the estimates conservative.

It's also a good idea to identify opportunities to increase sales leverage—that is, getting more yield from the same cost structure. How would it change the company if a sales rep could manage twice as many deals without slip-ups? Or if a sales manager could manage 50% more sales reps? How would the company operations change if the forecast were *really* reliable in week 10 of the quarter? These issues could mean better scalability for the company and ability to respond to market changes more quickly. These speed and scalability advantages increase both the profitability and the value of the company.

Beware of politics surrounding revenue estimates. It's tempting to say something like, "We could increase revenue 10% if only . . ." Sales executives may be very sensitive about this topic, and ornery finance executives might say, "We'll approve this project if only the Sales department is willing to increase its quota by 10%." To avoid this downhill spiral, focus your estimates on quantifying the *increased probability* of making the revenue targets the company already has. This approach is a less direct way of making the business case for SFDC, but it avoids the political traps and still makes the point.

Finally, beware of blatant double-counting of cost savings or revenue improvements. While some double-counting is almost unavoidable, if it's too obvious it simply lowers your credibility (note that the example in Table 1-2 shows just "the right amount" of double-counting).

Developing a Straw-Man Schedule

Part of making the business case for SFDC involves answering the question, "How soon can we get the benefits?" Once the project is approved, the schedule will be the single most visible aspect of managing the implementation. While SFDC and other SaaS systems can be turned on in a day or so (the SFDC sales reps will have made *sure* to tell you that), the standard system that is delivered will be of no practical use until it is configured (somehow the SFDC reps may not have emphasized this point). The time to configure the system may be short, but the time required to think through and test precisely how you need the system configured will be just as long as it would be with enterprise software. And the biggest cost and schedule elements—integration and data import—are just as large and uncertain as they would be with a conventional system. In large companies with a sophisticated IT department, allocate time in your schedule for initial review meetings: even though SFDC is a hosted system, security and compliance reviews will be required before you're allowed to "go live." Given that unexpected delays may be tagged as "project failures" by upper management, estimating the schedule and setting executive/sponsor expectations appropriately are key success factors for delivering the project.

As with any complex project, the SFDC schedule estimate has to be based on a reasonable best-case scenario: you assume that you won't make wrong decisions, that the right resources will be available when needed, and that there won't be any unpleasant discoveries along the way. To counteract this structural optimism, buffer elements must be added to the schedule during the riskiest tasks (such as development of a new software element or integrating an external system). Buffer elements must also be added during weeks that are critically constrained for sales personnel and executive management (e.g., sales managers are unlikely to be responsive to an action item during weeks 12 and 13 of the quarter, and no executive can be expected to attend an SFDC management review during the week of quarterly results and the board meeting). One of the first things a project manager should do is collect the calendar of "blackout periods" for each key resource. Immediately after completing this task, the project manager should identify "forcing functions"— external events with firm deadlines—that will likely drive the deployment

calendar. For example, an annual sales meeting will be an ideal time to roll out new functionality and do user training.

Gantt Charts, PERT Diagrams, and Other Management Delusions

While developing the schedule estimate, it really helps to use a Gantt charting tool such as Microsoft Project. This kind of software is fairly straightforward to learn, and during the planning phase its nice charts can help persuade busy managers about resource commitments.

By contrast, it is a very rare project that will actually *use* the Gantt charting tool during the life of the actual project, because maintaining the data and updating the schedule require a tremendous amount of effort. Unfortunately, few project participants will be willing to update their own information, so maintaining an accurate Gantt schedule will usually fall to the project manager. Most of the time, the day-to-day schedule will take the form of a spreadsheet. Set expectations with management accordingly.

That said, when a new project phase starts (or, less optimistically, when a major rework of the schedule is required), it is a good idea to redo the Gantt charts. Store backup copies of the original chart files for comparison purposes, but develop new ones that show the expected work breakdown structure at an overview level. You can have hours of fun comparing the original schedule with reality, and develop lessons learned for later phases of the project.

Ironically, one of the major areas of schedule uncertainty is in requirements setting and prioritization. A very common problem of large IT projects is to over-specify at two levels: at the high level, creating requirements that have questionable or unknown business value, and at the low level, specifying things in too much detail. At both levels, over-specification creates a lot of extra work and often engages employees in a level of perfectionism that has limited payoff.

The idea behind "value engineering" is making sure that a requirement is going to be *worth* satisfying before you spend the resources to do so. The best possible investment you can make as a business analyst or project manager is time spent vetting and validating requirements, because every hour you spend can save days of effort later. Allocate double the amount of time you think you'll need for interpreting and prioritizing requirements, as "smoking out" a marginal or bogus requirement can cut overall wasted effort in half.

Although the iterative process of an Agile project (described in Chapter 4) helps to contain the risk, any significant project will require a large number of meetings to discover what the requirements really are, how to prioritize and sequence them, and what they really mean for the system implementation.

The Art of the Quick Win

From the outset, it's important to develop a wide collection of user-visible features that can be liberally sprinkled through the schedule to maintain the perception that progress is being made and that the project is delivering business value on a frequent basis. This "quick win" list gives the project leader a tool to manage expectations and counteract a known negative (for example, "For the next two weeks, you can't do X, but we've given you this cool new feature Y to make your day go better"). Sometimes, a quick win is preventing a problem from ever occurring again. At other times, it's creating an alert to automatically flag unusual situations or adding a new "toy" that will keep users interested.

The quick win doesn't have to involve a lot of technology—the less, the better—and it should never involve a lot of work. It can be something as simple as a "mashup" with a traffic map so reps can figure out which route to drive to get to their appointments. Fortunately, SFDC's AppExchange has hundreds of free add-ons that provide features for users. Early in your project, do a survey of the AppExchange to see which freebies you can drop into the system during the implementation. Create a calendar of the "goody drops" and reschedule them as needed during the project

The project leader needs to creatively identify other opportunities for quick wins. Sometimes, just putting data in one place can create several opportunities for wins. For example, if you put all your employees' contact information into SFDC, which kind of reports could be generated? How about a mailing list for HR, or a birthday list for the administrative employees? These featurettes may take 10 minutes to implement, but each one can give you a week's worth of breathing room.

The sequence of phases can be as important as the amount of effort applied to the project, because the sequence of features determines which system benefits are visible to the users and sponsors. Unfortunately, even with SFDC some amount of effort will be required for enablers—infrastructure, integration, data cleansing, analysis, and testing—that don't deliver any specific feature (even though no feature would be possible without those functions). The resource for this infrastructure work needs to be planned first, before the visible-feature work is planned. However, always have *something* for the users. A project that devotes more than two thirds of the effort to things without visible results will have a tough time surviving in most organizations.

The project phases should start with the core of SFDC, and then add system extensions, external data, and external system integrations on top of a stable base. This order is preferred for two reasons: technical simplicity and user training. Users rarely have patience for training sessions lasting even an hour, so the amount of change you present to them during each session must be fairly small. You want users to become proficient with the core of the system that's immediately relevant to their jobs before you focus them on the fancier parts. Chapter 5 provides much more detail on training.

The early schedule for a deployment will look something like Table 1-3. Note that this schedule is **realistic only for a fairly small implementation of SFDC and a proficient implementation team.** Here are some guidelines to use as a sanity check on your schedule. Note that these recommendations are minimums, so it's fine if your schedule assumes slower progress. Many of these tasks can be carried out in parallel, so they may overlap on the calendar.

TABLE 1-3 Schedule for Small- to Medium-Sized Company SFDC Project Phases

Phase	Description	Start Date	Completion Date	Current Status	Business Processes	Affected Departments
1	Lead capture and routing	3-Mar	17-Mar	Done	Pipeline formation	Marketing, Sales
1	Campaigns and Web site integration	3-Mar	24-Mar	Done	Pipeline formation	Marketing
1	Historic lead import, 1 year	3-Mar	1-Apr	Done	Pipeline formation	Marketing
1	Lead scoring and qualification	18-Mar	7-Apr	Done	Pipeline formation	Marketing, Sales
2	Opportunity creation	8-Apr	15-Apr	Done	Pipeline formation	Sales
2	Basic forecasting reports	15-Apr	22-Apr	Done	Pipeline management	Sales
2	Opportunity aging and backtrack alcrts	15-Apr	29-Apr	Done	Pipeline management	Sales
2	Advanced forecasting	23-Apr	5-May	In progress	Pipeline management	Sales, Finance, Executives
2	Order-level quoting	1-May	15-May	In progress	Quote to invoice	Sales, Finance
2	Line-item-level quoting	16-May	1-Jun	Not started	Quote to invoice	Sales, Finance
3	Historic opportunity import, 1 year	16-May	15-Jun	Not started	Quote to invoice	Sales, Finance, Support
3	Historic opportunity import, 3 years	16-Jun	16-Aug	Not started	Quote to invoice	Sales, Finance, Support
3	Contract import, 1 year	16-May	15-Jun	Not started	Support and renewals	Sales, Finance, Support
3	Contract import, 3 years	16-Jun	16-Aug	Not started	Support and renewals	Sales, Finance, Support
3	Support ticket integration	16-May	1-Aug	Not started	Support and renewals	Support
3	Advanced forecasting reports and alerts	16-May	15-Jun	Not started	Pipeline management	Sales, Finance, Executives
3	Advanced campaign reports	16-May	15-Jun	Not started	Pipeline formation	Marketing
3	Dashboards	16-May	1-Jun	Not started	Executive management	Execs

- Initial SFDC bring-up, including basic configuration of `users, roles, profiles, territory assignments, security,` and basic user training: **2 person-weeks per 100 users or operating country.**
- Integration with your company's Web site and Web advertising campaigns: **1 person-week for a simple static system, 3 person-weeks for a full content-**

management system.

- Integration with external email blasting or marketing automation systems: **1 person-week per system,** but bugs and testing may make this much longer.
- Preparation for data import/migration: **1 person-day** on the learning curve for each data source.
- `Lead` information import, including data cleansing, deduping, reorganizing, and enrichment: **1,000–10,000 records per person-day,** depending on what kind of shape your data is in.[19]
- `Account` import, including account renaming and hierarchy creation, deduping, and enrichment: **500-5000 records per person day,** depending on how many sources of "accounts" you have and the shape of the data.
- `Contact` information import, including data cleansing, deduping, reorganizing, and enrichment: **500–5,000 records per person-day,** depending on what kind of shape your data is in.
- `Lead`/`contact activity` and `campaign history`, including data cleansing, deduping, reorganization, and enrichment: **500–5,000 records per person-day,** depending on what kind of shape your data is in.
- `Opportunity history` import, including creation of historical accounts, contacts, notes, and attachments as necessary, data cleansing, deduping, reorganizing, and enrichment: **40–5000 records per person-day,** depending on what kind of shape your data is in.
- `Price list`, product structures (SKUs), and quoting rules: **50–500 line items per person-day,** depending on the complexity of your products, pricing models, business rules, and update history. Due to the nature of price lists, integrating the full price list may require several weeks of additional analysis and restructuring.
- `Contract history` import, including data cleansing, deduping, reorganizing, and enrichment: **50–500 records per person-day,** depending on the shape your data is in. Many companies actually have to scan or key in data from paper contracts, which obviously means even lower productivity than indicated here.
- Customer `asset` inventory import (such as serial numbers, license keys, and other purchase history info): **500–5,000 records per person-day,** depending on what kind of shape your data is in.
- Support "ticket," "`case`," or "customer incident" data import, including creation of historical accounts, contacts, notes, and attachments as necessary, data cleansing, deduping, reorganizing, and enrichment: **40–5000 records per person-day,** depending on what kind of shape your data is in.
- Internal `reports` and `dashboards`: **1 person-week for the initial "toy"**

---

[19] Data need to be cleaned of errors (typos, bad entries), normalized (e.g., state names changed to ISO-standard codes), and validated (e.g., checking that the postal code corresponds with the state or province name). These tasks are trivial if all relevant data reside in one system and your processes inherently clean data as those data are used. Of course, the more data you have, the more different ways it can be messed up (particularly if the data have been imported from one system to another over several years). The worst-case scenario is having to cross-check system data with paper records or unstructured files (such as email, Word, Notes, or PDFs), as often happens with contracts. In these cases, it's important to import only the essential data, and to carefully assess the business value of processing another year's worth of historical records.

versions, although just the meetings to decide about the reports and modify them can take that long. Note that reports and dashboards will continue to evolve, so expect this effort to require several person-weeks of work over the first year of system use.

- External `reports`, data analysis, and business intelligence: **several person-weeks,** although it may take much longer depending on the amount of data to be analyzed and the tools used.
- `Workflows`, `alerts`, and basic automation: **1 person-week for the initial set,** although making decisions about policies and business rules can often take much longer. There is often a surprising lack of clarity about sales territories, compensation plans, business rules, and other automation essentials.
- Integration with external systems: no estimate is possible without a technical analysis of the two systems. Even though several SFDC partners provide "out of the box" connectors to scores of external software packages, in most cases **significant extra work is required** after the initial connector is installed. Often, the two systems cannot be perfectly synchronized, so extensive data manipulation may be required to reconcile the two systems' data sets.

It is almost impossible to budget too much time for data cleansing and integration projects. Don't try to be cheap—it will only hurt the project in the long run.

Avoiding the Big Bang Project

Throughout the process of "selling" the Salesforce.com project, getting the budget, and allocating the staff, expectations are being set about all the problems that will be solved. During this process, expectations are being set about the schedule as well—and often a fixed date (usually the first day of a fiscal quarter) emerges as an important corporate milestone. Finally, management develops expectations about the nature of the deployment, usually focusing on a system cutover, where users switch from the old way of doing things to the new system. And almost always, all of these expectations will **prove to be wrong.**

This story is as old as the computer industry itself. You don't have to believe me: Fred Brooks' classic *The Mythical Man Month* described how the larger the systems project, the more likely it was to be late. Even better, the further a project was into its schedule, the more likely it was that increasing the staffing and investment would make it even later.

The classic complete-system launch, sometimes called a Big Bang project, just doesn't work for software. It can work when there is military-style command and control: the Manhattan project, Operation Desert Storm, and the Apollo program are shining examples. But in business systems, it's hard to find examples where a Big Bang was a big success. Businesses don't have tight military hierarchies, and IT has a culture of trying to accommodate new business requirements during the project. This accommodation and tendency toward scope creep—particularly in software—is a poisonous cocktail. The Standish Group, an IT consultancy, published a series of studies that extensively analyzed failed IT projects. *The Chaos Reports* concluded that overblown requirements and overzealous adherence to large, monolithic deliveries were key warning signs of failed projects.

We'll discuss this issue in detail in Chapter 4. For now, be aware that the most important thing the project team can do before any project work begins is to **avoid three things:**

- Fake, vague, or overstated requirements
- Infrequent project milestones
- Large, complex, monolithic project deliverables

Avoiding Scope Creep

Scope creep is one of the most dangerous things that can happen to a project, and the danger grows in tandem with the size and length of the effort. This problem occurs whenever a requirement is stretched, an assumption made that "we can fit that in," or an *even better idea* is proposed by executives.

The chief symptom of scope creep: the requirements list grows while the project is under way. Because items aren't removed from the priority list, the number of deliverables grows even though the budget and schedule are unchanged (or worse, are being overspent by the project).

There's a particular temptation to load up the requirements even further whenever a project needs to get a schedule or budget extension, even though it should be obvious the project is already at risk of under-delivering.

The only solution for scope creep is vigilance among everyone on the project team, regular management reviews to root out new creepy requirements, a very persuasive project leader, and consistency on the part of executives. The project leader will need to develop political skills to escalate scope-creep issues without angering the people who are proposing the additional Great Things for the project.

Even if the project has been "sold" with a lofty set of goals and tight Big Bang deadlines, it is important to quickly move to a phased approach. This isn't just because of the IT "laws of physics"; it's also human nature. Put simply, change management and confidence building take time. For even small companies, reset executive expectations by making the following arguments:

- We all know what the desired end-state is going to be, and those goals have been agreed to. But until the project is fairly far along, we can't know which problems we'll find in our data, precisely how business processes need to change, or exactly what every user will need.
- Given this inevitable uncertainty, the best way to achieve our goals is to rapidly deliver a part of the project big enough to *provide value to the business*. This first delivery will give users a chance to learn the system and provide real-world feedback that will improve later system deliveries.
- Delivery after the first phase should be incremental, with each cycle providing a new element of value to the business. Having small, frequent deliveries makes it more likely we'll have happy users and low risk while meeting budgetary and schedule goals.

In making these arguments, the trick is to find the right "cornerstone functionality" to deploy for each phase, paired with the internal constituency you want to leverage. This decision making will

be profiled in detail for each phase as described in Chapter 4, but as part of selling the project you'll need to establish the constituency for the first phase's subsystem right now.

The best way to select the subsystem is to answer this question: Which system element could be deployed the soonest and deliver real value to a meaningful constituency? By producing a quick win, you earn credibility for the project as well as users for the system.

Outsourcing

Almost inevitably, you will need to use some outsource resources to assist in the project implementation, and you'll want at least budgetary placeholders for them. There are many tasks that your internal team ***shouldn't want to get good at.*** Even so, you cannot outsource everything—your team must be involved to define business rules, perform data extraction, review prototypes, validate data conversion, do acceptance testing, and, of course, manage the project. One thing to insist upon is a full "technology transfer" from any consultant you use, so that your team becomes self-sufficient in any area of ongoing development or maintenance.

In evaluating and choosing vendors, follow these rules of thumb:

- The product vendor's consultants will know more about the *internal* workings of their product than any partner consultant can. They will also have direct access to product engineering, and will have several "back door" tricks that can save time and make their prices a bargain. For internal extensions, customizations, scripting, rule setting, query design, and other "deep dive" projects directly related to their own products, the vendors' consultants are likely to be the best choice. However, they are less likely to be the right people for external integrations or business process design.
- A specialist system integrator with a practice dedicated to SFDC or other SaaS product is likely to know best how to do "external work" that integrates across applications. However, these individuals' level of knowledge and skill regarding business processes vary dramatically. In fact, many firms that are fully qualified at the technical level have zero capability when it comes to understanding and optimizing business processes. Check references before you make specific task allocations!
- A technical "coding shop" typically will be very good in its particular domain (for example, enhancing a CMS-driven Web site or an external order-management system), but those employees may not have experience with Web services and high-speed integration using SOAP.[20] Even though most SaaS applications use Web services for integration, every vendor uses the core technology a different way. For this reason, it's important to check the vendor's experience with SFDC's APIs and any AppExchange products you're planning to use. As always, references are more meaningful than vendor claims.

[20] Many readers won't know that SOAP is, but that's okay because it's a great litmus test for system integrators. Ask them to explain it to you: if they don't know that it stands for "Simple Object Access Protocol," or that SOAP is the way to join applications (or Web services) together in a lightweight, loosely coupled fashion, maybe you should look elsewhere. SFDC has a rich set of SOAP application programmer interfaces (APIs) that let you access and manipulate almost any system data from other applications. Ask would-be integrators which version of the APIs they've used (if they aren't up to version 14, they probably aren't up to snuff).

- A consultant who already works with your company has a natural advantage over outsiders. In addition to already being "connected" in your organization, the incumbent consultant is much more likely to know a lot about your technical environment and at least something about your business processes. Check with your IT organization to find out who is already under contract. If you've got only a few person-months of work to do, the incumbent's advantage may be a decisive one.
- Data cleansing and enrichment houses can be very economical (using tools and offshore resources unavailable to you), but this part of the project must be done *right and completely* to be at all meaningful. Check whether the vendor has done work on your size database in an SFDC project: most have not, and they'll experience a serious learning curve. Check references to confirm the details, because offshore data cleansing/enrichment operations tend to overstate their capabilities and are very *in*efficient at learning to do new things.
- Management consultants, sales process consultants, and organizational design specialists understand the human side of your business processes better than any other vendor type. Large organizations will typically need to use these consultants to optimize business processes, redesign sales procedures, and set policies. However, these consultants are rarely in a position to do implementation work, because they don't understand the inner workings of SFDC and other systems well enough. The also have a tendency to make recommendations that are technically very difficult or even unfeasible to implement. So absolutely include them on your team to design best practices and increase the speed of user adoption—but do not expect them to implement much in SFDC.

No matter which kinds of vendors you use, it is essential that you have a positive, cooperative relationship with them. Adversarial relationships just don't work—in terms of price, quality, or schedule. That said, it's essential that you manage each vendor relationship tightly. Although you don't manage a vendor in the same way you would an employee, vendors are every bit as important to project success and budget as any other team members. Weekly checkpoints and monthly scorecards are essential for all vendors.

Generally speaking, there are several areas that you *don't* want to outsource. It typically doesn't pay to have an outside team learn the peculiarities of your old home-grown systems. An outsourcer should not be making policy and process decisions for your company (they can make recommendations, but *you* have to own the decision).

Even if a major technology is implemented by outsiders, your internal team needs to have ongoing capability and knowledge. For example, you'll want to be able to make simple system modifications without calling in a vendor, and it's a good idea to be able to make minor changes to cope with the inevitable upgrades of external systems or minor updates to business rules. In your resource allocation, think through which of the tasks *must* be done internally, which should be done through a vendor, and which should be done jointly.

Setting Executive Expectations

Executives have been trained to see a business case that's firm and a schedule with a clear end-date. Unfortunately, large systems projects don't fit that model too often. And Agile project management, which is designed to make quality deliveries of what's really needed in the shortest

amount of time, creates a frustrating uncertainty about exactly what will be delivered when. The project leader must bridge that gap.

The first step is to convince the executives that SFA/CRM systems and processes must be adaptable to evolving business needs, so they are never completed the way a building would be. The next step is to focus attention on the cutover (or "go live") date, including the absolute minimum criteria that must be met to achieve this goal. Most significant SFDC implementations are replacements for previous systems, so it should be fairly straightforward to identify the criteria for the new system to be "good enough" to support the switchover. It is important to use terms like "good enough" or "acceptance criteria" to communicate that the system will continue to evolve after the go-live date. Some functional areas need to be only at the 80% level at cutover time to support the business.

As always, perfectionism does not pay. By focusing the discussion on what is really needed to support a given business process on a day-to-day basis, you can get realistic objectives and feedback on which tradeoffs could be acceptable at the go-live date. For example, it may be okay to *manually* approve orders for the first month of operation, so as to make sure that the automatic order approval rules accurately reflect your business policies.

The executives need to know that functionality will be delivered incrementally, and that their staff will need to play active roles in assuring quality over several phases. Executives also need to know when the key business processes affecting their specific departments will be done. Consequently, we recommend organizing the presentation of the schedule in two ways: chronologically, for the overview perspective, and organizationally, for each executive. Keeping the schedule in a spreadsheet (which can be generated from your Gantt charting tool) allows rapid creation of these executives' views, as shown in Figure 1-4. In this view, many tasks are repeated so that each executive can see the impact on his or her specific organization without having to refer elsewhere.

Phase	Description	Start Date	Completion Date	Current Status	Business Processes
Executives					
2	Advanced forecasting	23-Apr	5-May	In progress	Pipeline management
3	Advanced forecasting reports and alerts	16-May	15-Jun	Not started	Pipeline management
3	Dashboards	16-May	1-Jun	Not started	Executive management
Sales					
1	Lead capture and routing	3-Mar	17-Mar	Done	Pipeline formation
1	Lead scoring and qualification	18-Mar	7-Apr	Done	Pipeline formation
2	Opportunity creation	8-Apr	15-Apr	Done	Pipeline formation
2	Basic forecasting reports	15-Apr	22-Apr	Done	Pipeline management
2	Opportunity aging and backtrack alerts	15-Apr	29-Apr	Done	Pipeline management
2	Advanced forecasting	23-Apr	5-May	In progress	Pipeline management
2	Order-level quoting	1-May	15-May	In progress	Quote to invoice
2	Line-item-level quoting	16-May	1-Jun	Not started	Quote to invoice
3	Historic opportunity import, 1 year	16-May	15-Jun	Not started	Quote to invoice
3	Historic opportunity import, 3 years	16-Jun	16-Aug	Not started	Quote to invoice
3	Contract import, 1 year	16-May	15-Jun	Not started	Support and renewals
3	Contract import, 3 years	16-Jun	16-Aug	Not started	Support and renewals
3	Support ticket integration	16-May	1-Aug	Not started	Support and renewals
3	Advanced forecasting reports and alerts	16-May	15-Jun	Not started	Pipeline management
Marketing					
1	Lead capture and routing	3-Mar	17-Mar	Done	Pipeline formation
1	Campaigns and Web site integration	3-Mar	24-Mar	Done	Pipeline formation
1	Historic lead import, 1 year	3-Mar	1-Apr	Done	Pipeline formation
1	Lead scoring and qualification	18-Mar	7-Apr	Done	Pipeline formation
3	Advanced campaign reports	16-May	15-Jun	Not started	Pipeline formation
Finance					
2	Advanced forecasting	23-Apr	5-May	In progress	Pipeline management
2	Order-level quoting	1-May	15-May	In progress	Quote to invoice
2	Line-item-level quoting	16-May	1-Jun	Not started	Quote to invoice
3	Historic opportunity import, 1 year	16-May	15-Jun	Not started	Quote to invoice
3	Historic opportunity import, 3 years	16-Jun	16-Aug	Not started	Quote to invoice
3	Contract import, 1 year	16-May	15-Jun	Not started	Support and renewals
3	Contract import, 3 years	16-Jun	16-Aug	Not started	Support and renewals
3	Advanced forecasting reports and alerts	16-May	15-Jun	Not started	Pipeline management
Support					
3	Historic opportunity import, 1 year	16-May	15-Jun	Not started	Quote to invoice
3	Historic opportunity import, 3 years	16-Jun	16-Aug	Not started	Quote to invoice
3	Contract import, 1 year	16-May	15-Jun	Not started	Support and renewals
3	Contract import, 3 years	16-Jun	16-Aug	Not started	Support and renewals

Figure 1-4
Executive's Overview of Deployment Phases

One of the biggest challenges of setting executive expectations is the iceberg phenomenon: the toughest work in the project is in infrastructure and data cleanup that has no visible feature, no immediate "win" for any department. Most executives don't have patience for these intricacies of the project, and the best way to talk about them is in terms of "laying the foundation" for the features they want to see. Foundational work typically continues throughout the project, and it's often best to depict it during all phases. Even so, every single phase needs to deliver *some* interesting feature to at least one department, so the executives don't become frustrated with the amount of effort expended on "invisible stuff."

Getting the Right Resources Committed

At some point, there will be a meeting to make the decision to move forward. Usually, these meetings focus on the immediate expenditures. In contrast, commitments for ongoing expenses and effort are assumed away or quickly forgotten—and that's dangerous.

For a truly successful large-system implementation, the following funds must be assigned to the project *at the initial go/no-go meeting:*

- Initial procurement funding for the system, add-on products, and any hardware or IT resources required.
- Ongoing funding for the recurring fees, added to the budget for at least two years.
- Fees for consultants and service providers needed to configure, extend, integrate, and deploy the system.
- Fees for training courses and user-group sessions offered by the vendors.
- Dedicated personnel, typically in the form of a time allocation *with specific measured goals or MBOs:*
  - To make ongoing priority calls about requirements and schedules.
  - To decide policy issues and business rules.
  - To design approval cycles, exception handling, and workflows.
  - To do actual work on the SFDC system, including data cleansing, record imports, and other housekeeping tasks.
  - To do work on external systems such as: data modeling, data dumps, enabling external interfaces, doing testing, and other tasks.
  - To assist with IT infrastructure tasks (security audits, installing wiki software, server deployments, and so forth)
  - To do technical tests of the SFDC system and validation of its external integrations.
  - To do user testing of the system.
  - To run final acceptance tests.
- Regular (brief) intervals of executive time, to escalate issues, break logjams, reallocate resources, and do final approvals.

Surprisingly, most executives tend to focus on the top of this list. In reality, the items near the bottom tend to cause the biggest issues with projects over time.

At the meeting, you'll also need to set general expectations—preferably quantitative metrics of success and criteria for a successful deployment—for the first phase. Without these objective measurements, executives tend to remember only the date for the first phase deployment, which tends to put people in happy-ears mode. Make sure to start the project off on the right foot, with documented budgets, schedules, and success criteria.

