CHAPTER 5

Establishing the business requirements

Karen is a business analyst on a project to implement a new online product catalog for the company's customer service representatives. The drafted SRS is going through review when the marketing manager says he wants to add a "Like this product" feature. Karen's first instinct is to push back; there is already concern about meeting schedules with the current requirements set. But then she realizes that maybe that's a smart feature to add, because customer service representatives can promote the most-liked products with other customers. Before she elicits and documents functional requirements for this feature, she needs an objective analysis about whether this feature should be added to the scope or not.

When she explains to the marketing manager the need to analyze this request further, he responds, "Well, soon the developers are going to be in there changing code anyway. How hard is it to add just one tiny feature?" Karen's analysis determines that the proposed feature lies outside the project's scope: it won't contribute to the business objectives to reduce the customer service representatives' average call time, and it wouldn't be simple to implement. Karen needs to be able to clearly articulate why the feature isn't in scope to the marketing manager, who doesn't have the business objectives readily in mind.

As you saw in Chapter 1, "The essential software requirement," business requirements represent the top of the requirements chain. They define the vision of the solution and the scope of the project that will implement the solution. The user requirements and functional requirements must align with the context and objectives that the business requirements establish. Requirements that don't help the project achieve its business objectives shouldn't be implemented.

A project without a clearly defined and well-communicated direction invites disaster. Project participants can unwittingly work at cross-purposes if they have different objectives and priorities. The stakeholders will never agree on the requirements if they lack a common understanding of the project's business objectives. Without this understanding up front, project deadlines will likely be missed and budgets will likely be overrun as the team struggles to deliver the right product.

This chapter describes the vision and scope document, a deliverable that contains the project's business requirements. Figure 5-3 later in this chapter suggests a template for the vision and scope document. But before we get to the template, let's see just what we mean by "business requirements."

Defining business requirements

"Business requirements" refers to a set of information that, in the aggregate, describes a need that leads to one or more projects to deliver a solution and the desired ultimate business outcomes. Business opportunities, business objectives, success metrics, and a vision statement make up the business requirements.

Business requirements issues must be resolved before the functional and nonfunctional requirements can be fully specified. A statement of the project's scope and limitations helps greatly with discussions of proposed features and target releases. The business requirements provide a reference for making decisions about proposed requirement changes and enhancements. We recommend displaying the business objectives, vision, and scope highlights in every requirements elicitation session so the team can quickly judge whether a proposed requirement is in or out of scope.

Identifying desired business benefits

The business requirements set the context for, and enable the measurement of, the benefits the business hopes to achieve from undertaking a project. Organizations should not initiate any project without a clear understanding of the value it will add to the business. Set measurable targets with business objectives, and then define success metrics that allow you to measure whether you are on track to meet those objectives.

Business requirements might come from funding sponsors, corporate executives, marketing managers, or product visionaries. However, it can be challenging to identify and communicate the business benefits. Team members sometimes aren't exactly sure what the project is intended to accomplish. Sometimes, sponsors don't want to set objectives in a measurable fashion and then be held accountable for achieving them. There could be multiple important stakeholders who don't agree on what the objectives should be. The business analyst can ensure that the right stakeholders are setting the business requirements and facilitate elicitation, prioritization, and conflict resolution. Karl Wiegers (2006) suggests some questions that the BA can ask to help elicit business requirements.

The business benefit has to represent a true value for the project's sponsors and to the product's customers. For example, simply merging two systems into one is not a reasonable business objective. Customers don't care if they are using an application that involves 1, 5, or even 10 systems. They care about issues like increasing revenue and decreasing costs. Merging two systems might be part of the solution, but it is rarely the true business objective. Regulatory and legal compliance projects also have clear business objectives. Often the objectives are phrased as risk avoidance, possibly to avoid getting sued or being put out of business.

Product vision and project scope

Two core elements of the business requirements are the vision and the scope. The *product vision* succinctly describes the ultimate product that will achieve the business objectives. This product could serve as the complete solution for the business requirements or as just a portion of the solution. The vision describes what the product is about and what it ultimately could become. It provides the

context for making decisions throughout the product's life, and it aligns all stakeholders in a common direction. The *project scope* identifies what portion of the ultimate product vision the current project or development iteration will address. The statement of scope draws the boundary between what's in and what's out for this project.

Important The product vision ensures that we all know where we are hoping to go eventually. The project scope ensures that we are all talking about the same thing for the immediate project or iteration.

Make sure the vision solves the problem

In one of our training courses, we give students a business problem and a corresponding business objective. Throughout the exercise, we periodically provide additional details about the requirements. At each step, we ask the students to conceive a solution to the problem, given the information they have. By the end of the exercise, all of the students' solution ideas are similar, but rarely do any of them actually solve the original problem!

This mimics what we see on real projects. Teams might set clear objectives and then specify, develop, and test the system, without checking against the objectives along the way. A stakeholder might come up with a "shiny" new feature she wants implemented. The team adds it because it seems reasonable and interesting. However, months down the road, the delivered system doesn't solve the original problem, despite all of its cool features.

The vision applies to the product as a whole. The vision should change relatively slowly as a product's strategic positioning or a company's business objectives evolve over time. The scope pertains to a specific project or iteration that will implement the next increment of the product's functionality, as shown in Figure 5-1. Scope is more dynamic than vision because the stakeholders adjust the contents of each release within its schedule, budget, resource, and quality constraints. Scope for the current release should be clear, but the scope of future releases will be fuzzier the farther out you look. The team's goal is to manage the scope of a specific development or enhancement project as a defined subset of the strategic vision for the product.

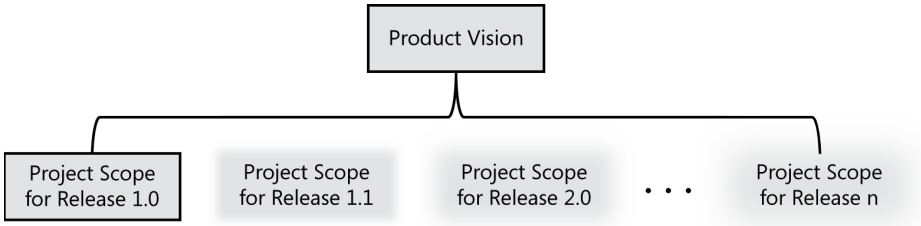

FIGURE 5-1 The product vision encompasses the scope for each planned release, which is less well defined the farther out you look.

Interlocking scopes

A federal government agency is undertaking a massive five-year information system development effort. The agency defined the business objectives and vision for this system early in the process; they won't change substantially over the next few years. The agency has planned 15 releases of portions of the ultimate system, each created by a separate project team and having its own scope description. Some projects will run in parallel, because certain of them are relatively independent of each other and some have longer timelines than others. Each scope description must align with the overall product vision and interlock with the scope for the other projects to ensure that nothing is inadvertently omitted and that lines of responsibility are clear.

Conflicting business requirements

Business requirements collected from multiple sources might conflict. Consider a kiosk that will be used by a retail store's customers. Figure 5-2 shows the likely business interests of the kiosk developer, retailer, and customer as we envision how each of these stakeholders hopes the kiosk will provide an advantage over their current way of doing business.

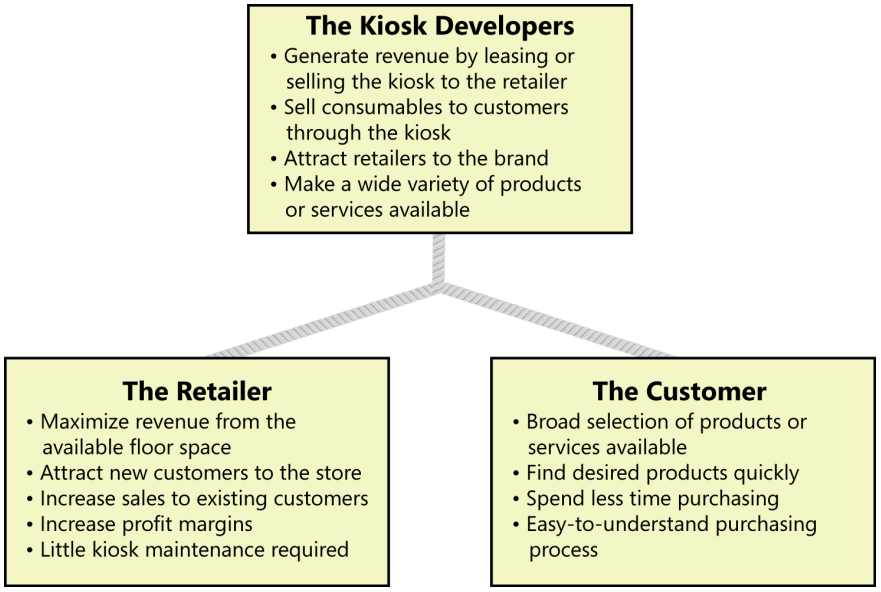

FIGURE 5-2 Stakeholders for a kiosk don't always have congruent business interests.

The various stakeholders' objectives sometimes are in alignment. For instance, both the kiosk developers and the customers want to have a wide variety of products or services available through the kiosk. However, some business objectives could conflict. The customer wants to spend less time purchasing goods and services, but the retailer would prefer to have customers linger in the store and

spend more money. The tension among stakeholders with different goals and constraints can lead to clashing business requirements. The project's decision makers must resolve these conflicts before the analyst can detail the kiosk's requirements. The focus should be on delivering the maximum business value to the primary stakeholders. It's easy to be distracted by superficial product characteristics that don't really address the business objectives.

The project's decision makers shouldn't expect the software team to resolve conflicts among various stakeholders. As more constituencies with diverse interests climb aboard, scope will grow. Uncontrolled scope creep, in which stakeholders overstuff the new system in an attempt to satisfy every interest, can cause the project to topple under its own weight. A BA can help by surfacing potential areas of conflict and differing assumptions, flagging conflicting business objectives, noting when requested features don't achieve those objectives, and facilitating conflict resolution. Resolving such issues is often a political and power struggle, which lies outside the scope of this book.

Long-duration projects often experience a change in decision makers partway through. If this happens to you, immediately revisit the baselined business requirements with the new decision makers. They need to be aware of the existing business requirements, which they might want to modify. If so, the project manager will have to adjust budgets, schedules, and resources, while the BA might need to work with stakeholders to update user and functional requirements and reset their priorities.

Vision and scope document

The *vision and scope document* collects the business requirements into a single deliverable that sets the stage for the subsequent development work. Some organizations create a project charter (Wiegers 2007) or a business case document that serves a similar purpose. Organizations that build commercial software often create a market (or marketing) requirements document (MRD). An MRD might go into more detail about the target market segments and the issues that pertain to commercial success.

The owner of the vision and scope document is the project's executive sponsor, funding authority, or someone in a similar role. A business analyst can work with this individual to articulate the business requirements and write the vision and scope document. Input to the business requirements should come from people who have a clear sense of why they are undertaking the project. These individuals might include the customer or development organization's senior management, a product visionary, a product manager, a subject matter expert, or members of the marketing department.

Figure 5-3 suggests a template for a vision and scope document; the sections that follow describe each of the template headings in more detail. As with any template, adapt this to meet the specific needs of your own projects. If you already have recorded some of this information elsewhere, do not duplicate it in the vision and scope document. Some elements of the vision and scope document might be reusable from project to project, such as business objectives, business risks, and stakeholder profiles. Appendix C includes an example vision and scope document written according to this template.

1. **Business requirements**
   - 1.1 Background
   - 1.2 Business opportunity
   - 1.3 Business objectives
   - 1.4 Success metrics
   - 1.5 Vision statement
   - 1.6 Business risks
   - 1.7 Business assumptions and dependencies
2. **Scope and limitations**
   - 2.1 Major features
   - 2.2 Scope of initial release
   - 2.3 Scope of subsequent releases
   - 2.4 Limitations and exclusions
3. **Business context**
   - 3.1 Stakeholder profiles
   - 3.2 Project priorities
   - 3.3 Deployment considerations

FIGURE 5-3 Suggested template for a vision and scope document.

The vision and scope document only defines the scope at a high level; the scope details are represented by each release baseline that the team defines. Major new projects should have both a complete vision and scope document and an SRS. (See Chapter 10, "Documenting the requirements," for an SRS template.) Each iteration, release, or enhancement project for an evolving product can include its own scope statement in that project's requirements documentation, rather than creating a separate vision and scope document.

Template tactics

Templates provide a consistent way to organize information from one project to the next. They help me remember information that I might overlook if I started with a blank piece of paper.

I don't fill out a template from top to bottom. Instead, I populate the various sections as I accumulate information during the course of the project. Empty sections highlight gaps in our current knowledge. Suppose one section of my document template is titled "Business risks." Partway through the project, I realize this section is empty. Does the project really have no business risks? Have we identified some business risks but stored them someplace else? Or have we not yet worked with appropriate stakeholders to identify possible risks? Blank sections in the template help me conduct a richer exploration for important project information. If there are common questions you ask to elicit content for a section, consider embedding those in the appropriate section of the template, perhaps in the form of hidden text, for others to reuse.

I use the term "shrink to fit" when working with templates. I begin with a rich template with many categories that might be important. Then I condense it down to just what I need for each situation. Suppose that a certain section of the template—business risks, say—doesn't pertain to the current project. I can remove that section from my document or I can retain the heading but leave the contents blank. Both options run the risk that a reader will notice the hole and

question whether there are indeed any business risks. The best solution is to put an explicit message in that section: "No business risks have been identified."

If certain sections of a template rarely get used, delete them. You might want to create a small set of templates for use on different types of projects, such as SRS templates suitable for use on large, new development projects; small websites; and enhancement projects. Even if you store your requirements in some repository other than a traditional document, a template can help you consider all the requirements information you need to accumulate for your project.

One project manager described the benefits his team received from adopting requirements document templates: "They are time consuming to fill in. The first couple of times I created them, I was surprised at the amount of detail required to make them useful, and then the amount of work taken to review and tidy up the documents, cleaning up any ambiguities, filling in gaps, etc. *But* it's worth it. The first two products that were developed after introducing the documents came in on time and were of much higher quality than before."

1. Business requirements

Projects are launched in the belief that creating or changing a product will provide worthwhile benefits for someone and a suitable return on investment. The business requirements describe the primary benefits that the new system will provide to its sponsors, buyers, and users. Business requirements directly influence which user requirements to implement and in what sequence.

1.1 Background

Summarize the rationale and context for the new product or for changes to be made to an existing one. Describe the history or situation that led to the decision to build this product.

1.2 Business opportunity

For a corporate information system, describe the business problem that is being solved or the process being improved, as well as the environment in which the system will be used. For a commercial product, describe the business opportunity that exists and the market in which the product will be competing. This section could include a comparative evaluation of existing products, indicating why the proposed product is attractive and the advantages it provides. Describe the problems that cannot currently be solved without the envisioned solution. Show how it aligns with market trends, technology evolution, or corporate strategic directions. List any other technologies, processes, or resources required to provide a complete customer solution.

Describe the needs of typical customers or of the target market. Present customer problems that the new product will address. Provide examples of how customers would use the product. Define any known critical interface or quality requirements, but omit design or implementation specifics.

1.3 Business objectives

Summarize the important business benefits the product will provide in a quantitative and measurable way. Platitudes ("become recognized as a world-class <whatever>") and vaguely stated improvements ("provide a more rewarding customer experience") are neither helpful nor verifiable. Table 5-1 presents some simplified examples of both financial and nonfinancial business objectives (Wiegers 2007).

TABLE 5-1 Examples of financial and nonfinancial business objectives

Financial	Nonfinancial
■ Capture a market share of X% within Y months. ■ Increase market share in country W from X% to Y% within Z months. ■ Reach a sales volume of X units or revenue of $Y within Z months. ■ Achieve X% return on investment within Y months. ■ Achieve positive cash flow on this product within Y months. ■ Save $X per year currently spent on a high-maintenance legacy system. ■ Reduce monthly support costs from $X to $Y within Z months. ■ Increase gross margin on existing business from X% to Y% within 1 year.	■ Achieve a customer satisfaction measure of at least X within Y months of release. ■ Increase transaction-processing productivity by X% and reduce data error rate to no more than Y%. ■ Develop an extensible platform for a family of related products. ■ Develop specific core technology competencies. ■ Be rated as the top product for reliability in published product reviews by a specified date. ■ Comply with specific federal and state regulations. ■ Receive no more than X service calls per unit and Y warranty calls per unit within Z months after shipping. ■ Reduce turnaround time to X hours on Y% of support calls.

Organizations generally undertake a project to solve a problem or exploit an opportunity. A business objectives model shows a hierarchy of related business problems and measurable business objectives (Beatty and Chen 2012). The problems describe what is keeping the business from meeting their goals at present, whereas the objectives define ways to measure achievement of those goals. Problems and objectives are intertwined: understanding one can reveal the other.

Given a set of business objectives, ask, "What is keeping us from achieving the goal?" to identify a more detailed business problem. Or work backward by asking, "Why do we care about that goal?" to better understand the top-level business problem or opportunity. Given a business problem, ask, "How will we assess whether the problem is solved?" to identify the measurable objective. The process is iterative, cycling through the hierarchy of problems and objectives until you see a list of features emerge that would help solve the problems and meet the objectives.

A conversation between a business analyst and an executive sponsor to identify business problems and objectives might look similar to the one in Figure 5-4. This illustration is for the Chemical Tracking System project at Contoso Pharmaceuticals that was introduced in Chapter 2, "Requirements from the customer's perspective." From the executive's responses to these questions, the BA could construct a business objectives model for the Chemical Tracking System, as shown in Figure 5-5.

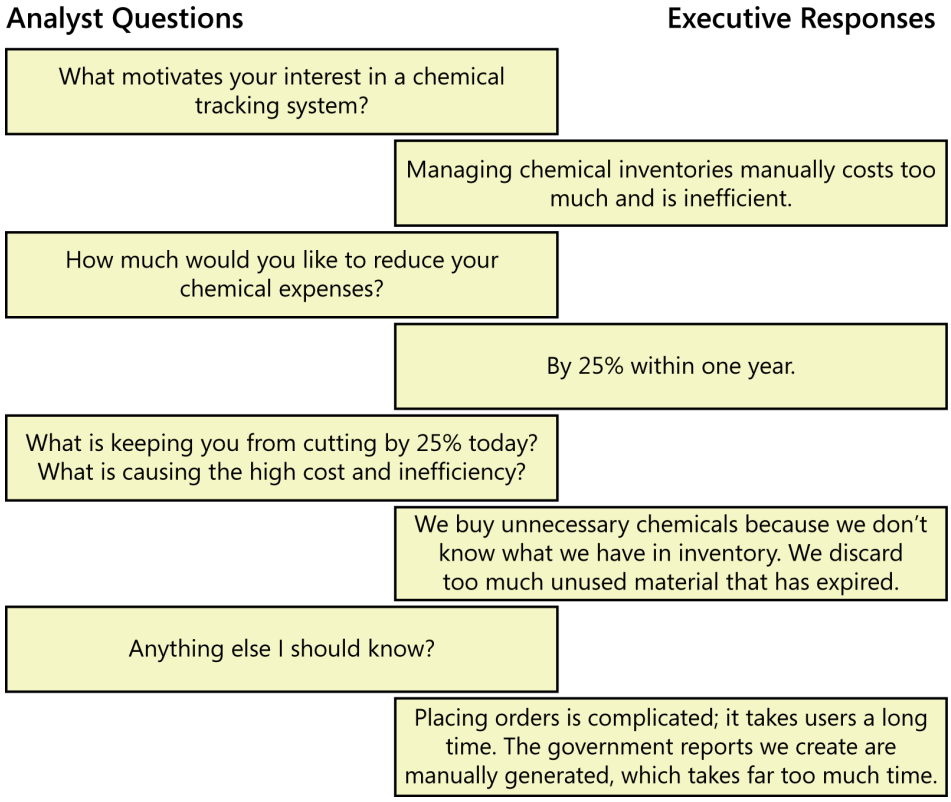

FIGURE 5-4 Example of a conversation between a business analyst and an executive sponsor.

1.4 Success metrics

Specify the indicators that stakeholders will use to define and measure success on this project (Wiegers 2007). Identify the factors that have the greatest impact on achieving that success, including factors both within and outside the organization's control.

Business objectives sometimes cannot be measured until well after a project is complete. In other cases, achieving the business objectives might be dependent on projects beyond your current one. However, it's still important to evaluate the success of an individual project. Success metrics indicate whether a project is on track to meet its business objectives. The metrics can be tracked during testing or shortly after product release. For the Chemical Tracking System, one success metric might be the same as Business Objective 3 in Figure 5-5 to "Reduce time spent ordering chemicals to 10 minutes on 80 percent of orders," because you can measure the average order time during testing or soon after release. Another success metric might relate to Business Objective 2 with a timeline that can be measured much earlier than a year after release, such as "Track 60 percent of commercial chemical containers and 50 percent of proprietary chemicals within 4 weeks."

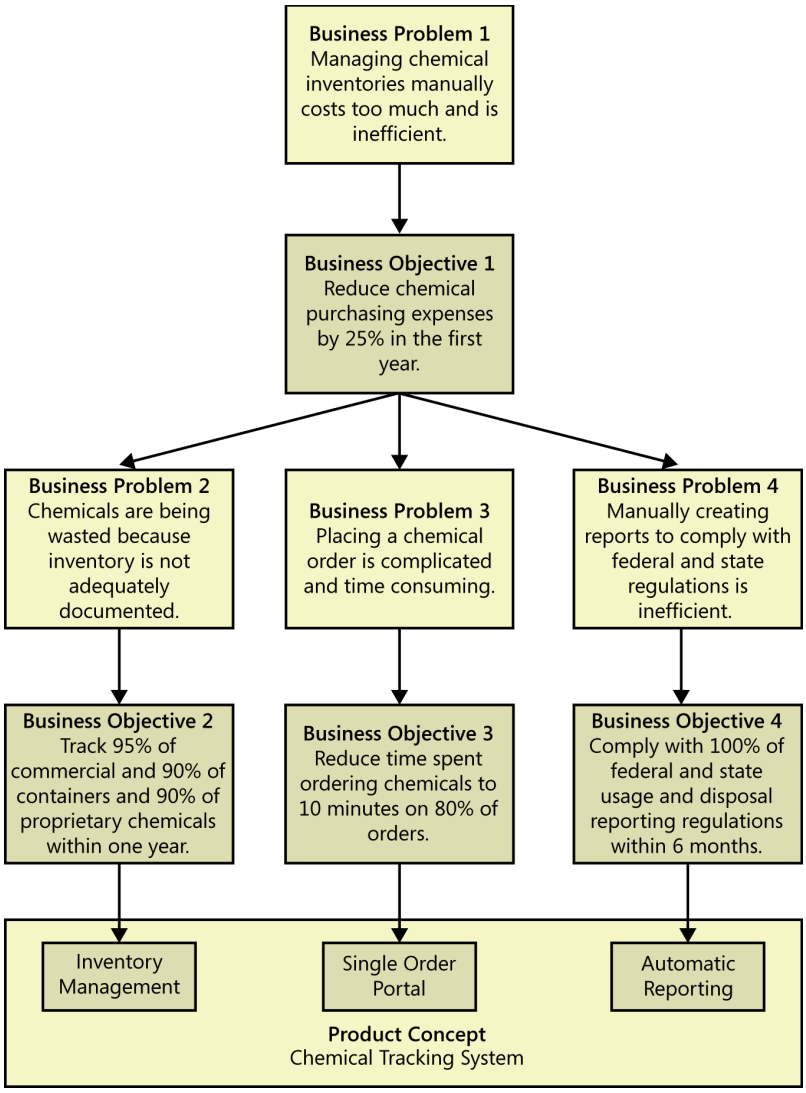

FIGURE 5-5 Example business objectives model for the Chemical Tracking System.

Important Choose your success metrics wisely. Make sure they measure what is important to your business, not just what is easy to measure. A success metric to "Reduce product development costs by 20 percent" is easy to measure. It might also be easy to achieve by laying off employees or investing less in innovation. However, these might not be the intended outcomes of the objectives.

1.5 Vision statement

Write a concise vision statement that summarizes the long-term purpose and intent of the product. The vision statement should reflect a balanced view that will satisfy the expectations of diverse stakeholders. It can be somewhat idealistic but should be grounded in the realities of existing or anticipated markets, enterprise architectures, corporate strategic directions, and resource limitations. The following keyword template works well for crafting a product vision statement (Moore 2002):

- ***For*** [target customer]
- ***Who*** [statement of the need or opportunity]
- ***The*** [product name]
- ***Is*** [product category]
- ***That*** [major capabilities, key benefit, compelling reason to buy or use]
- ***Unlike*** [primary competitive alternative, current system, current business process]
- ***Our product*** [statement of primary differentiation and advantages of new product]

Here's a sample vision statement for the Chemical Tracking System, with the keywords in boldface:

> ***For*** *scientists* ***who*** *need to request containers of chemicals,* ***the*** *Chemical Tracking System* ***is*** *an information system* ***that*** *will provide a single point of access to the chemical stockroom and to vendors. The system will store the location of every chemical container within the company, the quantity of material remaining in it, and the complete history of each container's locations and usage. This system will save the company 25 percent on chemical costs in the first year of use by allowing the company to fully exploit chemicals that are already available within the company, dispose of fewer partially used or expired containers, and use a standard chemical purchasing process.* ***Unlike*** *the current manual ordering processes,* ***our product*** *will generate all reports required to comply with federal and state government regulations that require the reporting of chemical usage, storage, and disposal.*

Crafting the product vision

I use the vision statement in my own consulting work. One longtime client and I work together very well, but occasionally Bill asks me to undertake a new project that's a little different. If we aren't exactly sure what he wants me to do, I ask him to write a vision statement. Bill always grumbles a bit because he knows that this will force him to think carefully about exactly what outcome he is expecting. But Bill's vision statement invariably gives me a clear idea of just what we are trying to accomplish so we can work efficiently together. It's well worth the time it takes.

You might have several key stakeholders write their vision statements separately, rather than doing it as a group exercise. Comparing their vision statements is a good way to spot different understandings about the project's objectives. And it's never too late to write a vision statement. Even if the project is under way, crafting a vision statement can help keep the rest of the project work on track and in focus. Though drafting a vision statement is quick, crafting the *right* vision statement and reaching agreement among the key stakeholders will take more time.

1.6 Business risks

Summarize the major business risks associated with developing—or not developing—this product. Risk categories include marketplace competition, timing issues, user acceptance, implementation issues, and possible negative impacts on the business. Business risks are not the same as project risks, which often include resource availability concerns and technology factors. Estimate the potential loss from each risk, the likelihood of it occurring, and any potential mitigation actions. See Chapter 32, "Software requirements and risk management," for more about this topic.

1.7 Business assumptions and dependencies

An *assumption* is a statement that is believed to be true in the absence of proof or definitive knowledge. Business assumptions are specifically related to the business requirements. Incorrect assumptions can potentially keep you from meeting your business objectives. For example, an executive sponsor might set a business objective that a new website will increase revenue by $100,000 per month. To establish this revenue target, the sponsor made some assumptions, perhaps that the new site will attract 200 additional unique visitors per day and that each visitor will spend an average of $17. If the new site does not attract enough visitors with a high enough average sale per visitor, the project might not achieve its business objective. If you learn that certain assumptions are wrong, you might have to change scope, adjust the schedule, or launch other projects to achieve the objectives.

Record any assumptions that the stakeholders made when conceiving the project and writing their vision and scope document. Often, one party's assumptions are not shared by others. If you write them down and review them, you can avoid possible confusion and aggravation in the future.

Record any major dependencies the project has on external factors. Examples are pending industry standards or government regulations, deliverables from other projects, third-party suppliers, or development partners. Some business assumptions and dependencies might turn into risks that the project manager must monitor regularly. Broken dependencies are a common source of project delays. Note the impact of an assumption not being true, or the impact of a broken dependency, to help stakeholders understand why it is critical.

2. Scope and limitations

When a chemist invents a new reaction that transforms one kind of chemical into another, he writes a paper that includes a "Scope and limitations" section, which describes what the reaction will and will not do. Similarly, a software project should define its scope and limitations. You need to state both what the solution being developed *is* and what it *is not*.

Many projects suffer from scope creep—rampant growth as more and more functionality gets stuffed into the product. The first step to controlling scope creep is to define the project's scope. The scope describes the concept and range of the proposed solution. The limitations itemize certain capabilities that the product will *not* include that some people might assume will be there. The scope and limitations help to establish realistic stakeholder expectations because customers sometimes request features that are too expensive or that lie outside the intended project scope.

Scope can be represented in numerous ways (see "Scope representation techniques" later in this chapter). At the highest level, scope is defined when the customer decides which business objectives to target. At a lower level, scope is defined at the level of features, user stories, use cases, or events and responses to include. Scope ultimately is defined through the set of functional requirements planned for implementation in a specific release or iteration. At each level, the scope must stay within the bounds of the level above it. For example, in-scope user requirements must map to the business objectives, and functional requirements must map to user requirements that are in scope.

Blue-sky requirements

A manager at a product development company that suffered near-catastrophic scope creep once told me ruefully, "We blue-skied the requirements too much." She meant that any idea anyone had was included in the requirements. This company had a solid product vision, but they didn't manage the scope by planning a series of releases and deferring some suggested features to later (perhaps infinitely later) releases. The team finally released an overinflated product after four years of development. It can be valuable to jot down the blue-sky requirements for future consideration. However, thoughtful scope management and an incremental development approach would have let the team ship a useful product much earlier.

2.1 Major features

List the product's major features or user capabilities, emphasizing those that distinguish it from previous or competing products. Think about how users will use the features, to ensure that the list is complete and that it does not include unnecessary features that sound interesting but don't provide customer value. Give each feature a unique and persistent label to permit tracing it to other system elements. You might include a feature tree diagram, as described later in this chapter.

2.2 Scope of initial release

Summarize the capabilities that are planned for inclusion in the initial product release. Scope is often defined in terms of features, but you can also define scope in terms of user stories, use cases, use case flows, or external events. Also describe the quality characteristics that will let the product provide the intended benefits to its various user classes. To focus the development effort and maintain a reasonable project schedule, avoid the temptation to include every feature that any potential customer might eventually want in release 1.0. Bloatware and slipped schedules are common outcomes of such insidious scope stuffing. Focus on those features that will provide the most value, at the most acceptable cost, to the broadest community, in the earliest time frame.

As an illustration, a recent project team decided that users had to be able to run their package delivery business with the first release of the software application. Version 1 didn't have to be fast, pretty, or easy to use, but it had to be reliable; this focus drove everything the team did. The initial release accomplished the basic objectives of the system. Future releases will include additional features, options, and usability aids. Be careful not to neglect nonfunctional requirements in the initial release, though. The ones that directly affect architecture are particularly critical to establish from the outset. Rearchitecting to try to fix quality deficiencies can be almost as expensive as a total rewrite. See Chapter 14, "Beyond functionality," for more about software quality attributes.

2.3 Scope of subsequent releases

If you envision a staged evolution of the product, or if you are following an iterative or incremental life cycle, build a release roadmap that indicates which functionality chunks will be deferred and the desired timing of later releases. Subsequent releases let you implement additional use cases and features, as well as enriching the capabilities of the initial ones. The farther out you look, the fuzzier these future scope statements will be and the more they will change over time. Expect to shift functionality from one planned release to another and to add unanticipated capabilities. Short release cycles provide frequent opportunities for learning based on customer feedback.

2.4 Limitations and exclusions

List any product capabilities or characteristics that a stakeholder might expect but that are not planned for inclusion in the product or in a specific release. List items that were cut from scope, so the scope decision is not forgotten. Maybe a user requested that she be able to access the system from her phone while away from her desk, but this was deemed to be out of scope. State that explicitly in this section: "The new system will not provide mobile platform support."

3. Business context

This section presents profiles of major stakeholder categories, management's priorities for the project, and a summary of some factors to consider when planning deployment of the solution.

3.1 Stakeholder profiles

Stakeholders are the people, groups, or organizations that are actively involved in a project, are affected by its outcome, or are able to influence its outcome (Smith 2000; IIBA 2009; PMI 2013). The stakeholder profiles describe different categories of customers and other key stakeholders for the project. You needn't describe every stakeholder group, such as legal staff who must check for compliance with pertinent laws on a website development project. Focus on different types of customers, target market segments, and the various user classes within those segments. Each stakeholder profile should include the following information:

- The major value or benefit that the stakeholder will receive from the product. Stakeholder value could be defined in terms of:
  - Improved productivity.

- Reduced rework and waste.
- Cost savings.
- Streamlined business processes.
- Automation of previously manual tasks.
- Ability to perform entirely new tasks.
- Compliance with pertinent standards or regulations.
- Improved usability compared to current products.

- Their likely attitudes toward the product.
- Major features and characteristics of interest.
- Any known constraints that must be accommodated.

You might include a list of key stakeholders by name for each profile or an organization chart that shows the relationships among the stakeholders within the organization.

3.2 Project priorities

To enable effective decision making, the stakeholders must agree on the project's priorities. One way to approach this is to consider the five dimensions of features, quality, schedule, cost, and staff (Wiegers 1996). Each dimension fits in one of the following three categories on any given project:

- **Constraint** A limiting factor within which the project manager must operate
- **Driver** A significant success objective with limited flexibility for adjustment
- **Degree of freedom** A factor that the project manager has some latitude to adjust and balance against the other dimensions

The project manager's challenge is to adjust the degrees of freedom to achieve the project's success drivers within the limits imposed by the constraints. Suppose marketing suddenly demands that you release the product one month earlier than scheduled. How do you respond? Do you:

- Defer certain requirements to a later release?
- Shorten the planned system test cycle?
- Demand overtime from your staff or hire contractors to accelerate development?
- Shift resources from other projects to help out?

The project priorities drive the actions you take when such eventualities arise. Realistically, when change happens, you need to have conversations with the key stakeholders to determine the most appropriate actions to take based on the change requested. For example, marketing might want to add features or shorten a timeline, but perhaps they are willing to defer certain features in exchange. See Appendix C for an example of how to document these project priorities.

Important Not all of the five dimensions can be constraints, and they cannot all be drivers. The project manager needs some degrees of freedom to be able to respond appropriately when requirements or project realities change.

3.3 Deployment considerations

Summarize the information and activities that are needed to ensure an effective deployment of the solution into its operating environment. Describe the access that users will require to use the system, such as whether the users are distributed over multiple time zones or located close to each other. State when the users in various locations need to access the system. If infrastructure changes are needed to support the software's need for capacity, network access, data storage, or data migration, describe those changes. Record any information that will be needed by people who will be preparing training or modifying business processes in conjunction with deployment of the new solution.

Scope representation techniques

The models described in this section can be used to represent project scope in various ways. You don't need to create all of these models; consider which ones provide the most useful insight for each project. The models can be included in the vision and scope document or stored elsewhere and referenced as needed.

The purpose of tools such as the context diagram, ecosystem map, feature tree, and event list is to foster clear and accurate communication among the project stakeholders. That clarity is more important than dogmatically adhering to the rules for a "correct" diagram. We strongly recommend, though, that you adopt the notations illustrated in the following examples as standards for drawing the diagrams. For example, in a context diagram, suppose you were to use a triangle to represent the system instead of a circle, and ovals rather than rectangles for external entities. Your colleagues would have difficulty reading a diagram that follows your personal preferences rather than a team standard.

Context diagrams, ecosystem maps, feature trees, and event lists are the most common ways to represent scope visually. However, other techniques are also used. Identifying affected business processes also can help define the scope boundary. Use case diagrams can depict the scope boundary between use cases and actors (see Chapter 8, "Understanding user requirements").

Context diagram

The scope description establishes the boundary and connections between the system you're developing and everything else in the universe. The *context diagram* visually illustrates this boundary. It identifies *external entities* (also called *terminators*) outside the system that interface to it in some way, as well as data, control, and material *flows* between the terminators and the system. The context diagram is the top level in a data flow diagram developed according to the principles of structured analysis (Robertson and Robertson 1994), but it's a useful model for all projects.

Figure 5-6 illustrates a portion of the context diagram for the Chemical Tracking System. The entire system is depicted as a single circle; the context diagram deliberately provides no visibility into the system's internal objects, processes, or data. The "system" inside the circle could encompass any combination of software, hardware, and human components. Therefore, it could include manual operations as part of the entire system. The external entities in the rectangles can represent user classes (Chemist, Buyer), organizations (Health and Safety Department), other systems (Training Database), or hardware devices (Bar Code Reader). The arrows on the diagram represent the flow of data (such as a request for a chemical) or physical items (such as a chemical container) between the system and its external entities.

You might expect to see chemical vendors shown as an external entity in this diagram. After all, the company will route orders to vendors for fulfillment, the vendors will send chemical containers and invoices to Contoso Pharmaceuticals, and Contoso's purchasing department will pay the vendors. However, those processes take place outside the scope of the Chemical Tracking System, as part of the operations of the purchasing and receiving departments. Their absence from the context diagram makes it clear that this system is not directly involved in placing orders with the vendors, receiving the products, or paying the bills.

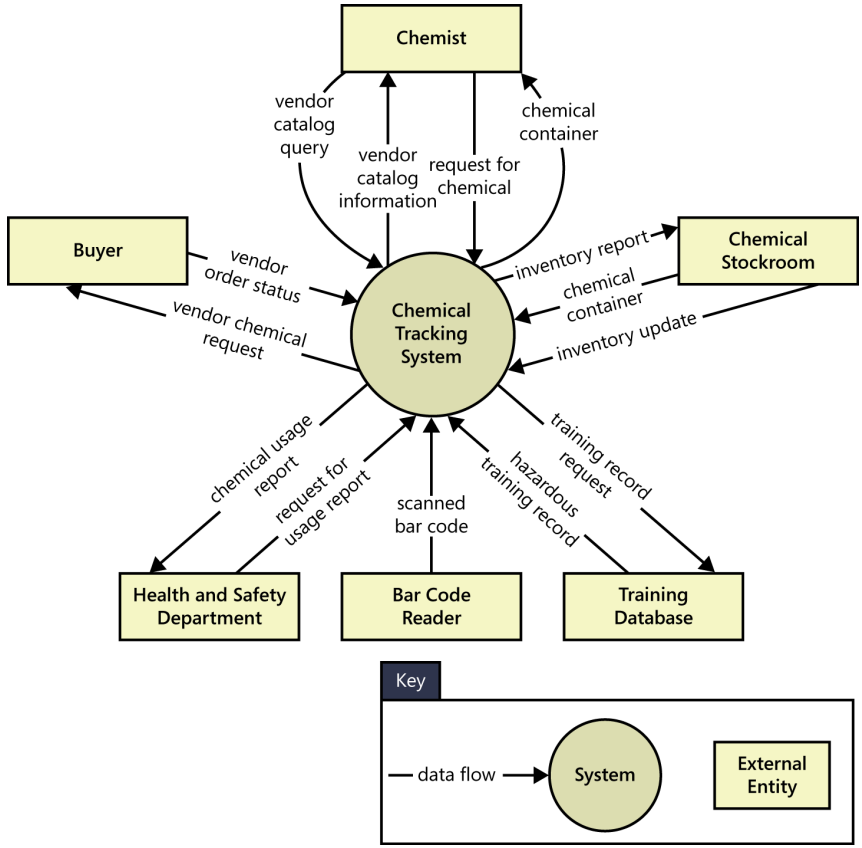

FIGURE 5-6 Partial context diagram for the Chemical Tracking System.

Ecosystem map

An *ecosystem map* shows all of the systems related to the system of interest that interact with one another and the nature of those interactions (Beatty and Chen 2012). An ecosystem map represents scope by showing all the systems that interconnect and that therefore might need to be modified to accommodate your new system. Ecosystem maps differ from context diagrams in that they show other systems that have a relationship with the system you're working on, including those without direct interfaces. You can identify the affected systems by determining which ones consume data from your system. When you reach the point that your project does not affect any additional data, you've identified the scope boundary of systems that participate in the solution.

Figure 5-7 is a partial ecosystem map for the Chemical Tracking System. The systems are all shown in boxes (such as the Purchasing System or Receiving System). In this example, the primary system we are working on is shown in a bold box (Chemical Tracking System), but if all systems have equal status in your solution, you can use the same box style for all of them. The lines show interfaces between systems (for instance, the Purchasing System interfaces to the Chemical Tracking System). Lines with arrows and labels show that major pieces of data are flowing from one system to another (for instance, "training records" are passed from the Corporate Training Database to the Chemical Tracking System). Some of these same flows can also appear on the context diagram.

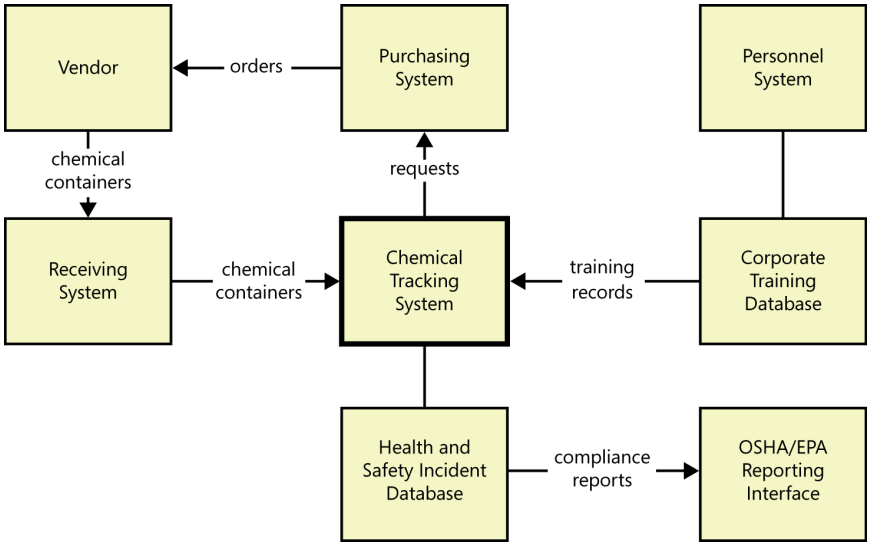

FIGURE 5-7 Partial ecosystem map for the Chemical Tracking System.

The ecosystem map in Figure 5-7 shows that the Chemical Tracking System does not directly connect to the OSHA/EPA Reporting Interface. Nonetheless, you need to consider whether any requirements in the Chemical Tracking System arise because of the data that flows from it, through the Health and Safety Incident Database, and to that reporting interface.

Feature tree

A *feature tree* is a visual depiction of the product's features organized in logical groups, hierarchically subdividing each feature into further levels of detail (Beatty and Chen 2012). The feature tree provides a concise view of all of the features planned for a project, making it an ideal model to show to executives who want a quick glance at the project scope. A feature tree can show up to three levels of features, commonly called level 1 (L1), level 2 (L2), and level 3 (L3). L2 features are subfeatures of L1 features, and L3 features are subfeatures of L2 features.

Figure 5-8 shows a partial feature tree for the Chemical Tracking System. The main branch of the tree in the middle represents the product being implemented. Each feature has its own line or "branch" coming off that central main branch. The gray boxes represent the L1 features, such as Order Chemicals and Inventory Management. The lines coming off an L1 branch are L2 features: Search and Chemical Request are subfeatures of Order Chemicals. The branches off an L2 branch are the L3 features: Local Lab Search is a subfeature of Search.

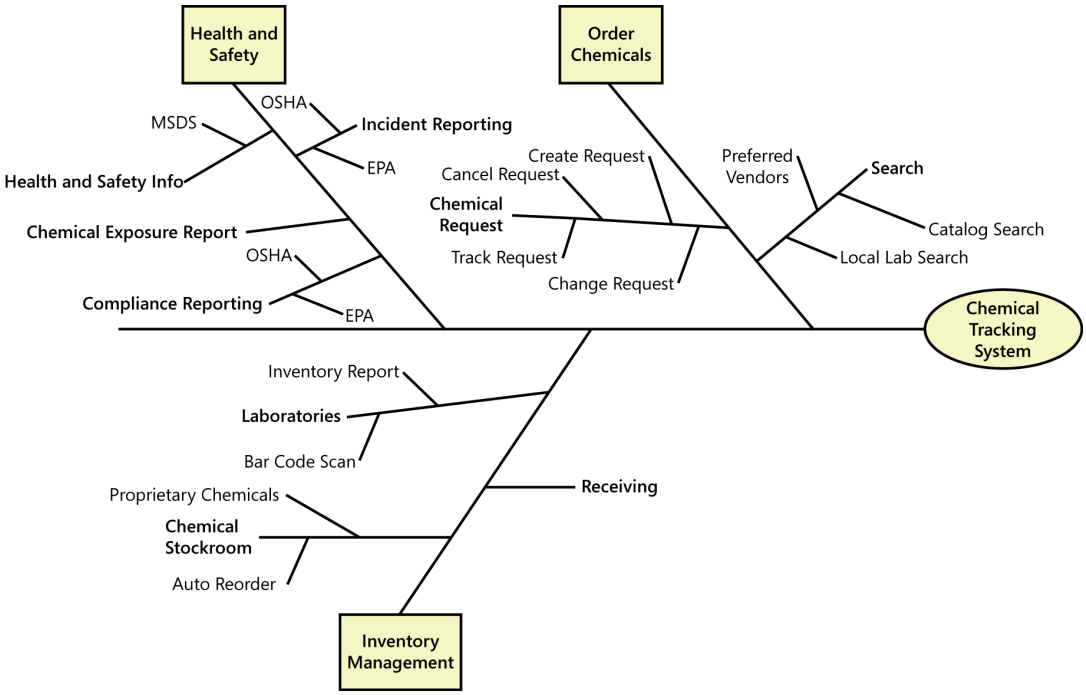

FIGURE 5-8 Partial feature tree for the Chemical Tracking System.

When planning a release or an iteration, you can define its scope by selecting a specific set of features and subfeatures to be implemented (Nejmeh and Thomas 2002; Wiegers 2006). You could implement a feature in its entirety in a specific release, or you could implement only a portion of it by choosing just certain L2 and L3 subfeatures. Future releases could enrich these rudimentary implementations by adding more L2 and L3 subfeatures until each feature is fully implemented in the final product. So the scope of a particular release consists of a defined set of L1, L2, and/or L3 features chosen from the feature tree. You can mark up a feature tree diagram to illustrate these feature

allocations across releases by using colors or font variations. Alternatively, you can create a feature roadmap table that lists the subfeatures planned for each release (Wiegers 2006).

Event list

An *event list* identifies external events that could trigger behavior in the system. The event list depicts the scope boundary for the system by naming possible business events triggered by users, time-triggered (temporal) events, or signal events received from external components, such as hardware devices. The event list only names the events; the functional requirements that describe how the system responds to the events would be detailed in the SRS by using event-response tables. See Chapter 12, "A picture is worth 1024 words," for more information about event-response tables.

Figure 5-9 is a partial event list for the Chemical Tracking System. Each item in the list states what triggers the event ("Chemist" does something or the "Time to" do something arrives), as well as identifying the event action. An event list is a useful scoping tool because you can allocate certain events to be implemented in specific product releases or development iterations.

External Events for Chemical Tracking System

- Chemist places a chemical request.
- Chemical container bar code is scanned.
- Time to generate OSHA compliance report arrives.
- Vendor issues new chemical catalog.
- New proprietary chemical is accessioned into system.
- Vendor indicates chemical is backordered.
- Chemist asks to generate his chemical exposure report.
- Updated material safety datasheet is received from EPA.
- New vendor is added to preferred vendor list.
- Chemical container is received from vendor.

FIGURE 5-9 Partial event list for the Chemical Tracking System.

Notice how the event list complements the context diagram and ecosystem map. The context diagram and ecosystem map collectively describe the external actors and systems involved, whereas the event list identifies what those actors and systems might do to trigger behavior in the system being specified. You can check the event list against the context diagram and ecosystem map for correctness and completeness, as follows:

- Consider whether each external entity on the context diagram is the source of any events: "Do any actions by Chemists trigger behavior in the Chemical Tracking System?"
- Consider whether any systems in the ecosystem map lead to events for your system.
- For each event, consider whether you have corresponding external entities in the context diagram or systems in the ecosystem map: "If a chemical container can be received from a vendor, does Vendor appear in the context diagram and/or ecosystem map?"

If you find a disconnect, consider whether the model is missing an element. In this case, Vendor did not appear on the context diagram because the Chemical Tracking System doesn't interface directly to vendors. However, Vendor is included in the ecosystem map.

Keeping the scope in focus

A scope definition is a structure, not a straitjacket. The business requirements and an understanding of how customers will use the product provide valuable tools for dealing with scope change. Scope change isn't a bad thing if it helps you steer the project toward satisfying evolving customer needs. The information in the vision and scope document lets you assess whether proposed requirements are appropriate for inclusion in the project. You can modify the scope for a future iteration or for an entire project if it's done consciously, by the right people, for the right business reasons, and with understanding and acceptance of the tradeoffs.

Remember, whenever someone requests a new requirement, the analyst needs to ask, "Is this in scope?" One response might be that the proposed requirement is clearly out of scope. Perhaps it's interesting, but it should be addressed in a future release or by another project. Another possibility is that the request obviously lies within the defined project scope. You can incorporate new in-scope requirements in the current project if they are of high priority relative to the other requirements that were already committed. Including new requirements often involves making a decision to defer or cancel other planned requirements, unless you're willing to extend the project's duration.

The third possibility is that the proposed new requirement is out of scope, but it's such a good idea that the scope should be broadened to accommodate it, with corresponding changes in budget, schedule, and/or staff. That is, there's a feedback loop between the user requirements and the business requirements. This will require that you update the vision and scope document, which should have been placed under change control at the time it was baselined. Keep a record of why requirements were rejected; they have a way of reappearing. Chapter 27, "Requirements management practices," describes how to use a requirement attribute to track rejected or deferred requirements.

Using business objectives to make scoping decisions

The business objectives are the most important factor to consider when making scope decisions. Determine which proposed features or user requirements add the most value with respect to the business objectives; schedule those for the early releases. When a stakeholder wants to add functionality, consider how the suggested changes will contribute to achieving the business objectives. For example, a business objective to generate maximum revenue from a kiosk implies the early implementation of features that sell more products or services to the customer. Glitzy features that appeal to only a few technology-hungry customers and don't contribute to the primary business objective shouldn't have high priority.

If possible, quantify the contribution the feature makes towards the business objectives, so that people can make scoping decisions on the basis of facts rather than emotions (Beatty and Chen 2012). Will a specific feature contribute roughly $1,000, $100,000, or $1,000,000 toward a business objective? When an executive requests a new feature that he thought of over the weekend, you can use quantitative analysis to help determine if adding it is the right business decision.

Assessing the impact of scope changes

When the project's scope increases, the project manager usually will have to renegotiate the planned budget, resources, schedule, and/or staff. Ideally, the original schedule and resources will accommodate a certain amount of change because of thoughtfully included contingency buffers (Wiegers 2007). Otherwise, you'll need to re-plan after requirements changes are approved.

A common consequence of scope change is that completed activities must be reworked in response to the changes. Quality often suffers if the allocated resources or time are not increased when new functionality is added. Documented business requirements make it easier to manage legitimate scope growth as the marketplace or business needs change. They also help a harried project manager to justify saying "no"—or at least "not yet"—when influential people try to stuff more features into an overly constrained project.

Vision and scope on agile projects

Managing scope on an agile project, in which development is performed in a series of fixed timebox iterations, takes a different approach. The scope of each iteration consists of user stories selected from a dynamic product backlog, based on their relative priority and the estimated delivery capacity of the team for each timebox. Instead of trying to fight scope creep, the team prioritizes new requirements against existing items in the backlog and allocates them to future iterations. The number of iterations—and hence the overall project duration—still depends on the total amount of functionality to be implemented, but the scope of each iteration is controlled to ensure timely completion. Alternatively, some agile projects fix the overall project duration, yet are willing to modify the scope. The number of iterations might remain the same, but the scope addressed in remaining iterations changes according to the relative priorities of existing and newly defined user stories.

The team can define a high-level roadmap of iterations at the beginning of the project, but the user story allocation for an iteration will be performed at the beginning of each iteration. Referencing the business requirements as the team sets the scope for each iteration helps to ensure that the project delivers a product that meets the business objectives. The same strategy can be used on any project that follows a timeboxed development process (see the "Scope management and timeboxed development" sidebar).

Scope management and timeboxed development

Enrique, a project manager at Litware, Inc., had to deliver a web-enabled version of Litware's flagship portfolio-management software. It would take about two years to fully supplant the mature application, but Litware needed a web presence right away. Enrique selected a timeboxed development approach, promising to release a new version every 90 days. His marketing team carefully prioritized the product's requirements. The SRS for each quarterly release included a committed set of new and enhanced features, as well as a list of lower-priority "stretch" requirements to be implemented as time permitted. Enrique's team didn't incorporate every stretch requirement into each release, but they did ship a stable release every three months through this schedule-driven approach to scope management. Schedule and quality are normally constraints on a timeboxed project, and scope is a degree of freedom.

Although agile projects might not create a formal vision and scope document, the contents from the template in Figure 5-3 are both relevant and essential to delivering a successful product. Many agile projects conduct an upfront planning iteration (iteration zero) to define the overarching product vision and other business requirements for the project.

Business requirements need to be defined for all software projects, regardless of their development approach. The business objectives describe the expected value coming out of the project, and on an agile project, they are used to help prioritize the backlog to deliver the most business value in the earliest iterations. Success metrics should be defined so that as iterative releases go live, the success can be measured and the rest of the backlog adjusted accordingly. A vision statement describes the long-term plan for what the product will be after all iterations are complete.

Using business objectives to determine completion

How do you know when you can stop implementing functionality? Traditionally, a project manager manages the project towards completion. However, a business analyst is intimately familiar with the business objectives and can help determine when the desired value has been delivered, implying that the work is done.

If you begin with a clear vision for the solution, and if each release or iteration is scoped to deliver just a portion of the total functionality, then you will be done when you complete the preplanned iterations. The completed iterations should have led to a fully realized product vision that meets the business objectives.

However, particularly in iterative development approaches, the end point might be vague. Within each iteration, scope is defined for that iteration. As the project continues, the backlog of uncompleted work dwindles. It's not always necessary to implement the entire set of remaining functionality. It's critical to have clear business objectives so that you can move toward satisfying those objectives incrementally as information becomes available. The project is complete when the success metrics indicate that you have a good chance of meeting the business objectives. Vague business objectives will guarantee an open-ended project with no way to know when you're done. Funding sponsors don't like it because they don't know how to budget, schedule, or plan for such projects. Customers don't like it because they might receive a solution that is delivered on time and on budget but that doesn't provide the value they need. But that might just be the risk of working on products that cannot be clearly defined at the outset, unless you refine the business objectives partway through the project.

Focus on defining clear business requirements for all of your projects. Otherwise, you are just wandering about aimlessly hoping to accomplish something useful without any way to know if you're reaching your destination.

Next steps

- Ask several stakeholders for your project each to write a vision statement using the keyword template described in this chapter. See how similar the visions are. Rectify any disconnects and come up with a unified vision statement that all those stakeholders agree to.
- Whether you're near the launch of a new project or in the midst of construction, document the business requirements by using the template in Figure 5-3. Or, simply create a business objectives model, and have the rest of the team review it. This might reveal that your team doesn't share a common understanding of the project's objectives or scope. Correct that problem now; it will be even more difficult to correct if you wait. This activity will also suggest ways to modify the template to best meet the needs of your organization's projects.
- Write down the measurable business objectives for your project in a format that can be shared easily in meetings throughout the project's duration. Take it to your next requirements-related meeting and see if the team finds the reminder to be useful.