

# The Requirements Process

# 2

*in which we present a process for  
discovering requirements and  
discuss how you might use it*



This book is a distillation of our experience. In it, we describe a requirements process that we have derived from our years of working in the requirements arena—working with clever people who do clever things, and working on projects in wonderfully diverse domains. We have also learned much from the experience of the many people around the world who use various parts of our techniques.

We developed the Volere Requirements Process and its associated specification template from the activities and deliverables that had proved themselves to be most effective in project and consulting assignments with our clients. The result of this experience is a requirements discovery and specification process whose principles can be applied—and indeed have been applied—to almost all kinds of application types in almost all kinds of development environments.

We want to stress from the very beginning that while we are presenting a process, we are using it as a vehicle for discovering requirements; we do *not* expect you to wave this process around and tell your co-workers that it is “the only way to do things.” However, we have high expectations that you will find many useful things from this process that will, in turn, help you to discover and communicate your requirements more productively and accurately. We have personally seen hundreds of companies adapt the process to their own cultures and organizations, and we know of thousands more that have done so.

Our clients who use the Volere Requirements Process are those who develop their products using RUP, incremental, iterative, spiral, Scrum, or other variations of iterative development; more formalized waterfall processes; and a variety of homebrewed development processes. Over the years,

---

*Whether you are building custom systems, building systems by assembling components, using commercial off-the-shelf software, accessing open-source software, outsourcing your development, or making changes to existing software, you still need to explore, discover, understand, and communicate the requirements.*

---

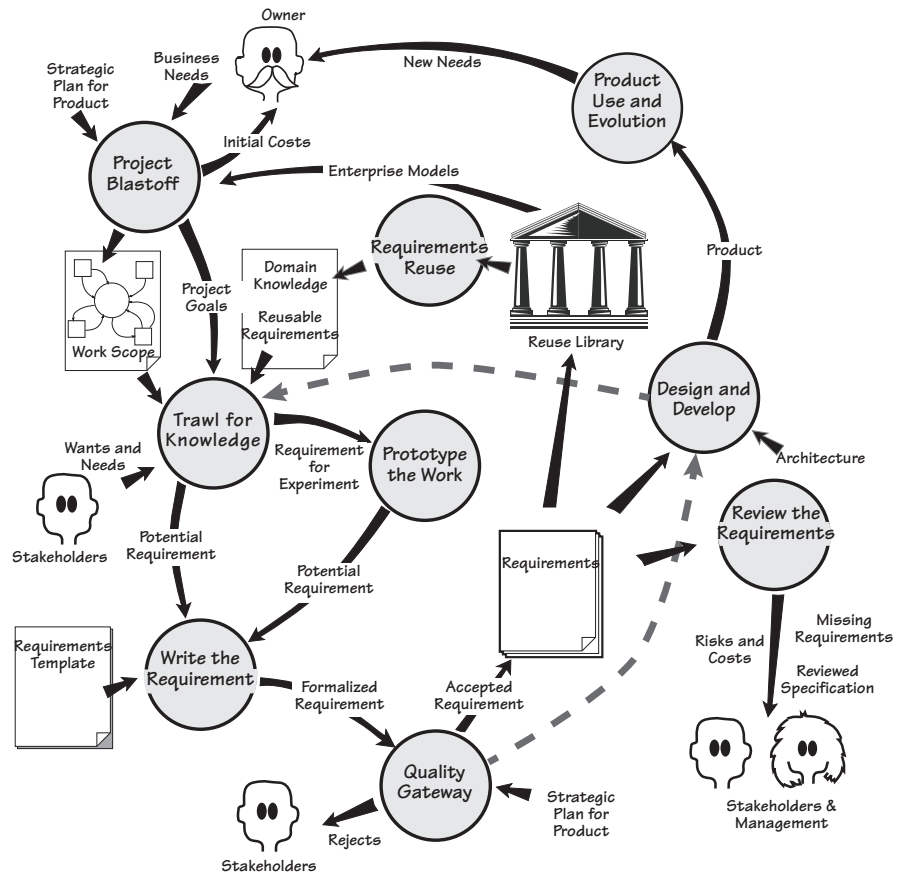
---

*If the right product is to be built, then the right requirements have to be discovered.*

---

**Figure 2.1**

This map of the Volere Requirements Process shows the activities and their deliverables. We have used a stylized data flow notation. Each activity (the bubbles) and its deliverables (named arrows or documents) are explained in the text. The dotted lines represent how this process is used with iterative projects.



all of these clients agreed with us: If the right product is to be built, the right requirements have to be discovered. But requirements don't come about by fortuitous accident. To find the correct and complete requirements, you need some kind of orderly process.

The Volere Requirements Process is shown in Figure 2.1. Each of the activities included in the figure, along with the connections between them, is described in detail in subsequent chapters of this book.

## The Requirements Process in Context

We need to point out—indeed, we need to stress—that this process is not intended to be a waterfall approach. At various stages throughout this book, we will point out how you might modify the process if you are using some kind of iterative development.

Requirements discovery should be seen as a necessary forerunner of any construction activity, but it should also be viewed as something that can be conducted quite quickly, sometimes quite informally, sometimes overlapping with subsequent design and construction activities, but never ignored.

Let's look briefly at each of the activities shown in Figure 2.1, which are covered in more detail in subsequent chapters. The intention of this chapter is to give you a gentle introduction to the process, its components, its deliverables, and the ways that they fit together. If you want more detail on any of the activities, feel free to jump ahead to the relevant chapter before completing this overview.

As we go through the process, we describe it as if you were working with a brand-new product—that is, developing something from scratch. We take this approach to avoid, for the moment, becoming entangled in the constraints that are part of all maintenance projects. Later, we will discuss requirements for those situations when the product already exists and changes to it are required.

## A Case Study

We will explain the Volere Requirements Process by taking you through a project that uses it.

The IceBreaker project is to develop a product that predicts when and where ice will form on roads, and to schedule trucks to treat the roads with de-icing material. The new product will enable road authorities to more accurately predict ice formation, schedule road treatments more precisely, and thereby make the roads safer. The product will also reduce the amount of de-icing material needed, which will help both the road authority's finances and the environment.

## Project Blastoff

Imagine launching a rocket. 10 – 9 – 8 – 7 – 6 – 5 – 4 – 3 – 2 – 1 – blastoff! If all it needed were the ability to count backward from 10, then even Andorra<sup>1</sup> would have its own space program. The truth of the matter is that before we get to the final 10 seconds of a rocket launch, a lot of preparation has taken place. The rocket has been fueled, and the course plotted—in fact, everything that needs to be done if the rocket is to survive and complete a successful mission.

The key purpose of the project blastoff is to build the foundation for the requirements discovery that is to follow, and to ensure that all the needed components for a successful project are in place. The principal stakeholders—the sponsor, the key users, the lead requirements analyst, technical and business experts, and other people who are crucial to the success of the project—gather together to arrive at a consensus on the crucial project issues.

“The likelihood of frost or ice forming is determined by the energy receipt and loss at the road surface. This energy flow is controlled by a number of environmental and meteorological factors (such as exposure, altitude, road construction, traffic, cloud cover, and wind speed). These factors cause significant variation in road surface temperature from time to time and from one location to another. Winter night-time road surface temperatures can vary by over 10°C across a road network in a county.”

—Vaisala News

**Blastoff is also known as “project initiation,” “kickoff,” “charter,” “project launch,” and many other things. We use the term “blastoff” to describe what we are trying to achieve—getting the requirements project launched and flying.**

### FOOTNOTE 1

Andorra is a tiny principality in the Pyrenees mountains between France and Spain. Only since 1993 has it been

a parliamentary democracy, but it retains its ancient chiefs of state as a coprincipality. The responsibilities of the French prince are now vested with the president of France. On the Spanish side, the “prince” is the bishop of Seo de Urgel.

Andorra became famous in the 1960s for having a defense budget of \$4.50, a tale that has become the stuff of legend. Today Andorra’s defense budget is zero.

Refer to Chapter 3, Scoping the Business Problem, for a detailed discussion of project blastoff.

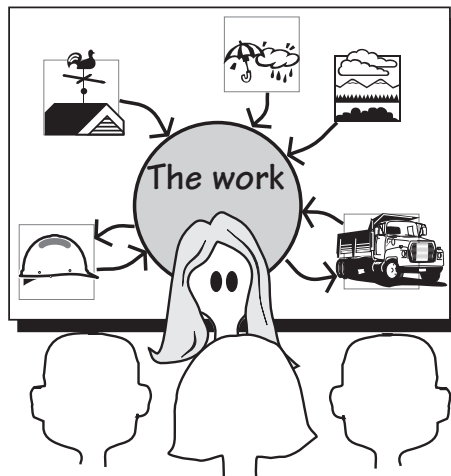
The blastoff defines the scope of the business problem and seeks concurrence from the stakeholders that yes, this is the area of the owner’s organization that needs to be improved. The blastoff meeting confirms the functionality to be included in the requirements discovery, and the functionality that is to be specifically excluded.

Defining the scope of the business problem is usually the most convenient way to start. In the IceBreaker project, the lead requirements analyst coordinates the group members’ discussion as they come to a consensus on the scope of the work—that is, the business area to be improved—and how this work relates to the world around it. The meeting participants draw a *context diagram* on a whiteboard to show which functionality is included in the work, and by extension, which elements they consider to be outside the scope of the ice forecasting business. The diagram defines—precisely defines—the included functionality by showing the connections between the work and the outside world. (More on this in the next chapter.) This use of a context diagram is illustrated in Figure 2.2. Later, as the requirements activity proceeds, the context diagram is used to reveal the optimal product to help with this work.

When they have reached a reasonable agreement on the scope of the business area to be studied, the group identifies the *stakeholders*. The stakeholders are those people who have an interest in the product, or who have knowledge pertaining to the product—in fact, anyone who has requirements for it. For the IceBreaker project, the people who have an interest are the road engineers, the truck depot supervisor, the weather forecasting people, road safety experts, ice treatment consultants, and so on. These people must be identified, so that the requirements analysts can work with them to find all the requirements. The context diagram, by establishing the extent of the work, helps to identify many of the stakeholders.

**Figure 2.2**

The context diagram is used to build a consensus among the stakeholders as to the scope of the work that needs to be improved. The eventual product will be used to do part of this work.



The blastoff also confirms the *goals* of the project. The blastoff group comes to an agreement on the business reason for doing the project, and agrees that there is a clear and measurable benefit to be gained by doing the project. The group also agrees that the product is worthwhile for the business to make the investment, and that the organization is capable of building and operating it.

It is sensible project management practice at this stage to produce a preliminary estimate of the costs involved for the requirements part of the project—this can be done by using the information already contained in the context diagram. It is also sensible project management to make an early assessment of the risks that the project is likely to face. Although these risks might seem like depressing news, it is always better to get an idea of the downside of the project (its risk and cost) before being swept away by the euphoria of the benefits that the new product is intended to bring.

The blastoff group members arrive at a consensus on whether the project is worthwhile and viable—that is, they make the “go/no go” decision. It might seem brutal to kill off an embryonic project, but we know from bitter experience that it is better to cancel a project at an early stage than to have it stagger on for months—or years—consuming valuable resources when it has little or no chance of success. The blastoff group carefully considers whether the product is viable, and whether its benefits outweigh its costs and risks.

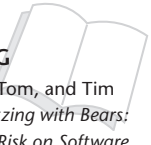
Alternatively, if too many unknowns remain at this point, the blastoff group might decide to start the requirements investigation with the intention of reviewing the requirements after a short while and reassessing the value of the project.

---

*It is always better to get an idea of the downside of the project (its risk and cost) before being swept away by the euphoria of the benefits that the new product is intended to bring.*

---

#### READING



DeMarco, Tom, and Tim Lister. *Waltzing with Bears: Managing Risk on Software Projects*. Dorset House, 2003.


McConnell, Steve. *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006.

## Trawling for Requirements


Once the blastoff is completed, the business analysts start *trawling* the work to learn and understand its functionality—“What’s going on with this piece of the business, and what do they want it to do?” For convenience and consistency, they partition the work context diagram into business use cases.

Each business use case is an amount of functionality needed by the work to make the correct response to a business event. (These terms will be fully explained soon.) A requirements analyst is assigned to each of the business use cases—the analysts can work almost independently of one another—for further detailed study. The analysts use trawling techniques such as apprenticing, scenarios, use case workshops, and many others to discover the true nature of the work. These trawling techniques are described in Chapter 5, *Investigating the Work*.

Trawling means discovering the requirements. The business analysts sit with the IceBreaker technicians as they describe the work they currently do, and their aspirations for work they hope to do. The business analysts



Refer to Chapter 4 for a discussion of business events and business use cases, and an exploration of how you might use them.



Refer to Chapter 5, *Investigating the Work*, for details of the trawling activity.

We look at developing innovative products in Chapter 8, Starting the Solution.

### READING

Maiden, Neil, Suzanne Robertson, Sharon Manning, and John Greenwood. *Integrating Creativity Workshops into Structured Requirements Processes*. Proceedings of DIS 2004, Cambridge, Mass. ACM Press.

Michalko, Michael. *Thinkertoys: A Handbook of Creative-Thinking Techniques*, second edition. Ten Speed Press, 2006.

Robertson, Suzanne, and James Robertson. *Requirements-Led Project Management*. Addison-Wesley, 2005.

also consult with other interested stakeholders and subject-matter experts—experts on usability, security, operations, management, and so on—to discover other needs for the eventual product. The IceBreaker business analysts spent a lot of time with the meteorologists and the highway engineers.

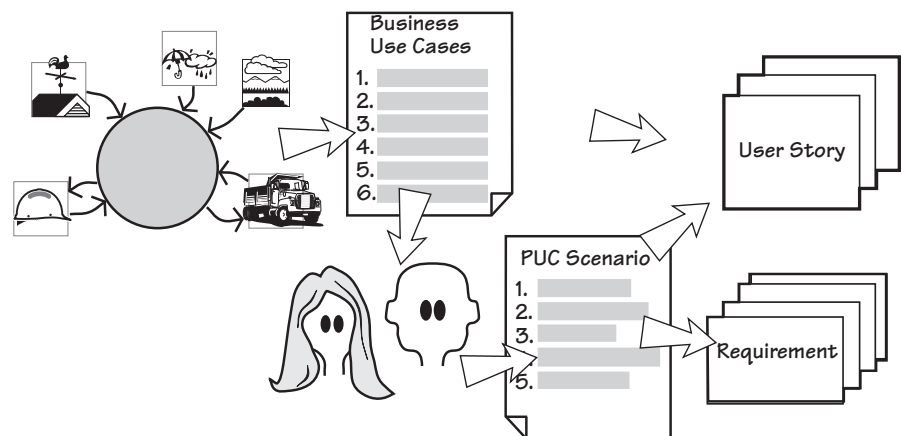
Perhaps the most difficult part of requirements investigation is uncovering the *essence* of the system. Many stakeholders inevitably talk about their perceived *solution* to the problem or express their needs in terms of the current implementation. The essence, by contrast, is the underlying business reason for having the product. Alternatively, you can think of it as the *policy* of the work, or what the work or the business rule would be if it could exist without any technology (and that includes people). We will have more to say about the essence of the system in Chapter 7, Understanding the Real Problem.

Once they understand the essence of the work, the analysts get together with the key stakeholders to decide the best product to improve this work. That is, they determine how much of the work to automate or change, and what effect those decisions will have on the work. Once they know the extent of the product, the requirements analysts write its requirements. We illustrate this process in Figure 2.3.

The IceBreaker product must not be a simplistic automation of the work as it is currently done; the best of our automated products are not mere imitations of an existing situation. To deliver a truly useful product, the analytical team must work with the stakeholders to innovate—that is, to develop a better way to do the work, and a product that supports this better way of working. They make use of innovation workshops where the team uses creative thinking techniques and innovative triggers to generate new and better ideas for the work and the eventual product.

Figure 2.3

The blastoff determines the scope of the work to be improved. The business use cases are derived from the scope. Each of the business use cases is studied by the requirements analysts and the relevant stakeholders to discover the desired way of working. When this is understood, the appropriate product can be determined (the PUC scenario) and requirements or user stories written from it.

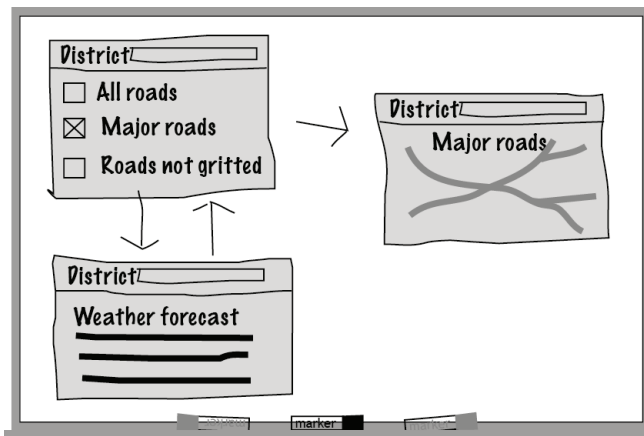


## Quick and Dirty Modeling

Models can be used at any time in the Volere life cycle; in Figure 2.1, we show this activity as “Prototype the Work.” There are, of course, formal models such as you would find in UML or BPMN, but a lot of the time business analysts can make productive use of quick sketches and diagrams to model the work being investigated. One quick and dirty modeling technique we should mention here is using Post-it notes to model functionality; each note can be used to represent an activity, and the notes can be rapidly rearranged to show different ways the work is done or could be done. We find that stakeholders relate to this way of modeling their business processes, and are always willing to participate with hands-on manipulation of the Post-its to show what they think the work should be. We discuss this kind of modeling more fully in Chapter 5, *Investigating the Work*.

In Chapter 8, *Starting the Solution*, we examine how you move into an implementation of the requirements discovered so far. At this point, your models change from being something to explain the current work, to something to explain how the future product will help with that work.

We can now start to refer to this type of model as a prototype—a quick and dirty *representation* of a potential product using pencil and paper, whiteboards, or some other familiar means, as shown in Figure 2.4. Prototypes used at this stage are intended to present the user with a simulation of the requirements as they might be implemented. The IceBreaker business analysts sketch some proposed interfaces and ways that the needed functionality might be implemented—this visual way of working allows the engineers and other stakeholders to coalesce their ideas for the future product.



**Figure 2.4**

A quick and dirty prototype built on a whiteboard to provide a rapid visual explanation of how some of the requirements might be implemented, and to clarify misunderstood or missing requirements.



## Scenarios

Scenarios are so useful that we have devoted the whole of Chapter 6 to them. Scenarios show the functionality of a business process by breaking it into a series of easily recognizable steps, written in English (or whatever language you use at work) so that they are accessible to all stakeholders. The IceBreaker analysts used scenarios to describe the business processes and present their understanding of the needed functionality. These scenarios were then revised as needed—different stakeholders took an interest in different parts of the scenario, and after a short time, the business analysts were able to have everyone understand and come to a consensus on what the work was to be.

Once they are agreed, the scenarios become the foundation for the requirements.

Refer to Chapter 6 for a discussion about using scenarios.

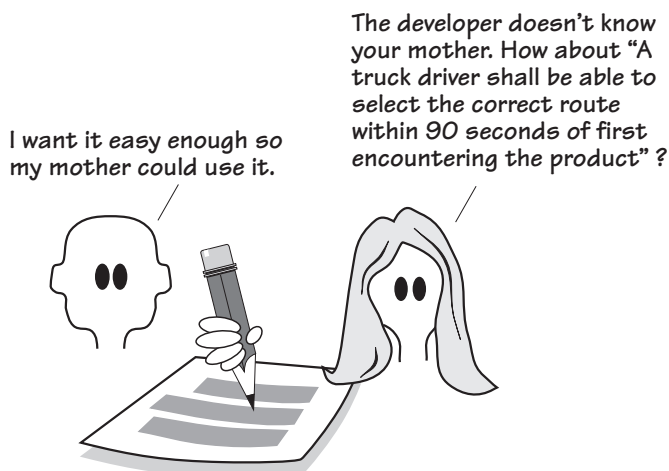
## Writing the Requirements

A major problem in system development is misunderstood requirements. To avoid any misunderstanding, the analysts must write their requirements in an unambiguous and testable manner, and at the same time ensure that the originating stakeholder understands and agrees with the written requirement before it is passed on to the developers. In other words, the analysts write the requirements so as to ensure that parties at either end of the development spectrum are able to have an identical understanding of what is needed.

Although the task of writing down the requirements might seem an onerous burden, we have found it to be the most effective way to ensure that the essence of the requirement has been captured and communicated, and that the delivered product can be tested. (See Figure 2.5.)

**Figure 2.5**

The requirements are captured in written form to facilitate communication between the stakeholders, the analysts, and the developers (and anyone else who has an interest). By writing the requirements carefully, the team ensures that the correct product is built.





The IceBreaker analysts start by writing their requirements using business language so that the nontechnical stakeholders can understand them and verify their correctness. They add a *rationale* to the requirements—it shows the background reason for the requirement, which removes much of the ambiguity. Further, to ensure complete precision and to confirm that the product designers and developers can build exactly what the stakeholder needs, they write a *fit criterion* for each requirement. A fit criterion quantifies, or measures, the requirement, which makes it testable, which in turn allows the testers to determine whether an implementation meets—in other words, fits—the requirement.

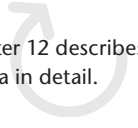
The rationale and the fit criterion make the requirement more understandable for the business stakeholder, who has on several occasions said, “I am not going to have any requirements that I do not understand, nor will I have any that are not useful or that don’t contribute to my work. I want to understand the contributions that they make. That’s why I want each one to be both justified and measurable.”

The business analyst has a different, but complementary, reason for measuring requirements: “I need to ensure that each requirement is unambiguous; that is, it must have the same meaning to both the stakeholder who originated it and the developer who will build it. I also need to measure the requirement against the stakeholder’s expectations. If I can’t put a measurement to it, then I can never tell if we are building the product the stakeholder really needs.”

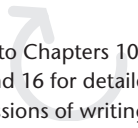
The analysts use two devices to make it easier to write their specification. The first device, the *requirements specification template*, is an outline and guide to writing a requirements specification. The business analysts use it as a checklist of the requirements they should be asking for, and as a consistent way of organizing their requirements documents. The second device is a *shell*, also known as a *snow card*. Each atomic (that’s the lowest level) requirement is made up of a number of attributes, and the snow card is a convenient layout for ensuring that each requirement has the correct constituents.

Of course, the writing process is not really a separate activity. In reality, it is integrated with the activities that surround it—trawling, prototyping, and the quality gateway. However, for the purposes of understanding what is involved in putting the correct requirements into a communicable form, we will look at it separately.

Iterative development methods employ *user stories* as a way of conveying the requirements. The stories are, in fact, placeholders for lower-level requirements; they are augmented during conversations between the developers and the stakeholders to flush out the detailed requirements. In Chapter 14, Requirements and Iterative Development, we look closely at how the business analyst can produce better user stories. Working iteratively does not obviate the need for requirements, but rather seeks to discover and communicate the requirements in a different manner.



Chapter 12 describes fit criteria in detail.



Refer to Chapters 10, 11, 12, and 16 for detailed discussions of writing the requirements.

The primary reason for wanting written requirements is not to *have* written requirements (although that is often necessary), but rather to *write* them. Writing the requirement, together with its associated rationale and fit criterion, clarifies it in the writer's mind, and sets it down in an unambiguous and verifiable manner. To put that another way, if the business analyst cannot correctly write the requirement, he has not yet understood it.

## Quality Gateway

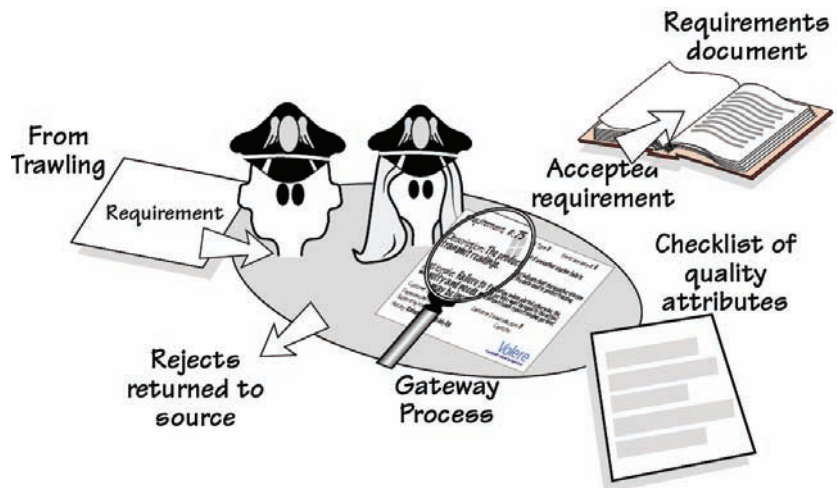
Requirements are the foundation for all that is to follow in the product development cycle. Thus it stands to reason that if the right product is to be built, the requirements must be correct before they are handed over to the builders. To ensure correctness, the quality gateway tests the requirements (Figure 2.6). The IceBreaker team has set up a single point that every requirement must pass through before it can become a part of the specification. This gateway is manned by two people—the lead requirements analyst and a tester—and they are the only people authorized to pass requirements through the gateway. Working together, they check each requirement for completeness, relevance, testability, coherency, traceability, and several other qualities before they allow it to be passed to the developers.

By ensuring that the only way for requirements to be made available for the developers is for those requirements to pass through the quality gateway, the project team is in control of the requirements, and not the other way around.

Chapter 13 describes how the quality gateway tests the requirements.

**Figure 2.6**

The quality gateway ensures that requirements are rigorous by testing each one for completeness, correctness, measurability, absence of ambiguity, and several other attributes, before allowing the requirement to be passed to the developers.



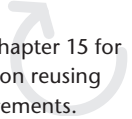
## Reusing Requirements

The requirements for any product you build are never completely unique. We suggest that before starting on any new requirements project, you go through the specifications written for previous projects and look for potentially reusable material. Sometimes you may find dozens of requirements that you can reuse without alteration. More often you will find requirements that, although they are not exactly what you want, are suitable as the basis for some of the requirements you will write in the new project.

For example, in the IceBreaker project, the rules for road engineering have not changed much over the years. Thus, the requirements analysts working on various projects do not have to rediscover them, but can simply reuse them. They also know that the business of vehicle scheduling does not change radically over time, so their trawling process can take advantage of some requirements from previous projects.

Similarly, for different projects within your organization, the non-functional requirements are fairly standard, so you can start with a specification from one of the previous projects and use it as a checklist.

The point about reusing requirements is that once a requirement has been successfully specified for a product, and the product itself is successful, the requirement does not have to be reinvented or rediscovered. In Chapter 15, Reusing Requirements, we discuss how you can take advantage of the knowledge that already exists within your organization, and how you can save yourself time by recycling requirements from previous projects.

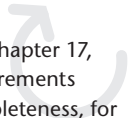


See Chapter 15 for more on reusing requirements.

## Reviewing the Requirements

The quality gateway exists to keep bad requirements out of the specification—it does this one requirement at a time. Nevertheless, at the point when you think your requirements specification is complete (or as complete as you need it for the next activity), you should review it. This final review checks that there are no missing requirements, that all the requirements are consistent with one another, and that any conflicts between the requirements have been resolved. In short, the review confirms that the specification is really complete and suitable so that you can move on to the next stage of development.

This review also offers you an opportunity to reassess the costs and risks of the project. Now that you have a complete set of requirements, you know a lot more about the product than you did at the project blastoff. In particular, you have a much more precise knowledge of the scope and functionality of the product, so this is a good time to remeasure its size. From that size, and from your knowledge of the project's constraints and solution architecture, you can estimate the cost to construct the product.



See Chapter 17, Requirements Completeness, for more on reviewing the specification.

You also know at this stage which types of requirements are associated with the greatest risks. For example, the users might have asked for an interface that your organization has not built before. Or perhaps they want to use untried technology to build the product. Perhaps the developer might not have the people with the skills needed to build the product as specified? By reassessing the risks at this point, you give yourself a more realistic chance of building the desired product successfully.

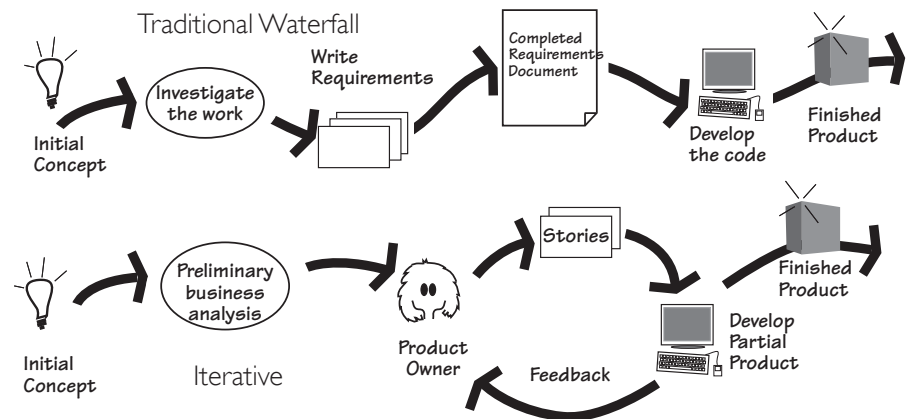
## Iterative and Incremental Processes

One common misconception in the requirements world is that you have to gather *all* the requirements before moving on to the next step of design and construction. In other words, doing requirements means that you employ a traditional waterfall process. In some circumstances this is necessary, but not always. On the one hand, if you are outsourcing or if the requirements document forms the basis of a contract, then clearly you need to have a complete requirements specification. On the other hand, if the overall architecture is known, then construction and delivery can often begin before all the requirements are discovered. We show these two approaches in Figure 2.7, and suggest you consider which one works best for you when working on your own requirements projects. We also have a lot more to say on various approaches in Chapter 9, Strategies for Today's Business Analyst.

On the IceBreaker project, the developers are ready to start building the product, so after the blastoff the key stakeholders select three (it could be any low number) of the highest-priority and greatest-value business use cases. The requirements analysts trawl and gather the requirements for only those

**Figure 2.7**

Two (of many) variations on development life cycles. At the top of the figure is the traditional waterfall approach, in which the complete requirements document is put together before product development begins. At the bottom of the figure is an iterative process, in which, after a preliminary analysis, the product is developed in small increments. Both approaches achieve the same purpose.



business use cases, putting aside the rest of the work for now. Then, when the first tranche of requirements have successfully passed the quality gateway, the developers start their work. The intention is to implement a small number of use cases as early as possible to get the reaction of the stakeholders—if there are going to be any nasty surprises, the IceBreaker team wants to get them as early as possible. While the developers are building and delivering the first lot of business use cases, the analysts are working on the requirements for the next-highest-priority ones. Soon they have established a rhythm for delivery, with new use cases being implemented and delivered every few weeks.

## Requirements Retrospective

You are reading this book about a requirements process, presumably with the intention of improving your own process. Retrospectives, sometimes known as *lessons learned*, are one of the most effective tools for discovering the good and bad of a process, and suggesting remedial action. Retrospectives for requirements projects consist of a series of interviews with stakeholders and group sessions with the developers. The intention is to canvas all the people involved in the project and ask these questions:

- What did we do right?
- What did we do wrong?
- If we had to do it again, what would we do differently?

By looking for honest answers to these questions, you give yourself the best chance of improving your process. The idea is very simple: Do more of what works and less of what doesn't.

Keep a record of the lessons learned from your retrospectives. While humans have memory and can learn from their experience to their advantage in future projects, organizations don't learn—unless you write down the experience. By keeping the lessons learned available in some readily accessible manner, subsequent projects can learn from your accomplishments and mishaps.

Your retrospective can be very informal: a coffee-time meeting with the project group, or the project leader collecting e-mail messages from the participants. Alternatively, if the stakes are higher, this process can be formalized to the point where it is run by an outside facilitator who canvases the participants, both individually and as a group, and publishes a retrospective report.

The most notable feature of retrospectives is this: Companies that regularly conduct retrospectives consistently report significant improvements in their processes. In short, retrospectives are probably the cheapest investment you can make in improving your own process.

---

*"If we did the project again tomorrow, what would we do differently?"*

---

## Evolution of Requirements

You start a project with little more than a vision—and sometimes a fairly blurred vision—of the desired future state of your owner’s work. (As we have done elsewhere in this book, we use the term “work” to refer to the area of the owner’s organization where improvements are to be made, usually by automating or re-automating part of it.)

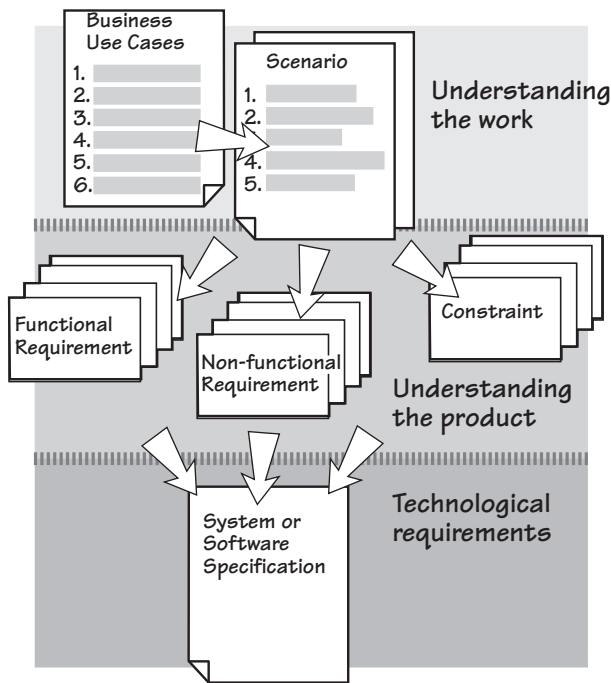
During the early stages of requirements discovery, analysts deploy models of varying degrees of formality to help them and the stakeholders to learn what the work is, and what it is to be. From this investigation of the work, everyone arrives at the same level of understanding such that the stakeholders find improvements that will be truly beneficial.

It helps enormously when coming to an understanding of the work if the analysts and stakeholders can see the *essence* of the work. The essence is an abstraction of the work that sees the underlying policy of the work without the technology that clouds our vision of what the work actually is. This “thinking above the line,” as we call it in Chapter 7, Understanding the Real Problem, is important if the requirements are not to merely replicate whatever it is that exists at the moment, and if “technological fossils” and inappropriate process are not to be inadvertently reimplemented.

The understanding of the work evolves and matures, and at some point it is possible for the stakeholders, guided by the business analysts and the systems architects, to determine the optimal product to improve that work. When this stage is reached, the business analysts determine the detailed functionality for the product (keep in mind that not all of the work’s functionality would be included in the product) and to write its requirements. The non-functional requirements are derived at roughly the same time and written along with those constraints that are not already recorded. At this point, the requirements are written in a technologically neutral manner—they specify what the product has to do for the work, but not how the technology will do it.

You can think of these requirements as “business requirements,” meaning that they specify the product needed to support the business. Once they are adequately understood, they are released to the designer, who adds the product’s technological requirements before producing the final specification for the builders. This process is illustrated in Figure 2.8.

We have said that the requirements evolve, but this process should not be thought of as an inexorable progression toward some known destination. As Earl Beede points out, every time you think of a solution, it causes some new problems that require you to backtrack and revisit some of your earlier work. When we are talking about a requirements process, keep in mind that the process, if it is to be useful, must allow you to move backward as well as forward. Naturally, you would like to spend most of your time moving

**Figure 2.8**

The requirements evolve as development of the product progresses. They start out as fairly vague ideas as the analysts and stakeholders explore the work area. As the ideas for the product emerge over time, the requirements become precise and testable. They remain technologically neutral until the designer becomes involved and adds those requirements needed to make the product work in its technological environment.

forward, but don't be too disappointed if you have to return to some things you thought you had put behind you.

## The Template

It is easier to write requirements, and far more convenient, if you have a guide to writing them. Appendix A of this book provides The Volere Requirements Specification Template, which is a complete blueprint for describing your product's functionality and capabilities. This template, which is a distillation of literally hundreds of requirements specifications, is in use by thousands of organizations all over the world.

It is convenient to categorize requirements into several types—each of the template's sections describes a type of requirement and its variations. Thus, as you discover the requirements with your stakeholders, you add them to your specification, using the template as a guide to necessary content.

The template is designed to serve as a sophisticated checklist, providing you with a list of what to write about, and suggestions on how to write about them. The table of contents for the template is reproduced here, and we will discuss each section in detail later in the book.

Our associate, Stephen Mellor, suggests using the template by going directly to the most pressing sections—the ones that seem to you to be most

The complete Volere Requirements Specification Template is found in Appendix A.



useful—and then revisiting the template as needed. You will probably use most of it, but it is not—really not—a template that you fill by starting on page one and working through to the bitter end. Like any good tool, when used wisely the template provides a significant advantage to your requirements discovery.

Here, then, is the content of the template.

### **Project Drivers**—reasons and motivators for the project

- 1 **The Purpose of the Project**—the reason for making the investment in building the product and the business advantage that you want to achieve by doing so
- 2 **The Client, the Customer, and Other Stakeholders**—the people with an interest in or an influence on the product
- 3 **Users of the Product**—the intended end users, and how they affect the product's usability

### **Project Constraints**—the restrictions on the project and the product

- 4 **Requirements Constraints**—the limitations on the project, and the restrictions on the design of the product
- 5 **Naming Conventions and Definitions**—the vocabulary of the project
- 6 **Relevant Facts and Assumptions**—outside influences that make some difference to this product, or assumptions that the developers are making

### **Functional Requirements**—the functionality of the product

- 7 **The Scope of the Work**—the business area or domain under study
- 8 **The Scope of the Product**—a definition of the intended product boundaries and the product's connections to adjacent systems
- 9 **Functional and Data Requirements**—the things the product must do and the data manipulated by the functions

### **Non-functional Requirements**—the product's qualities

- 10 **Look and Feel Requirements**—the intended appearance
- 11 **Usability and Humanity Requirements**—what the product has to be if it is to be successfully used by its intended audience
- 12 **Performance Requirements**—how fast, big, accurate, safe, reliable, robust, scalable, and long-lasting, and what capacity

- 13 **Operational and Environmental Requirements**—the product’s intended operating environment
- 14 **Maintainability and Support Requirements**—how changeable the product must be and what support is needed
- 15 **Security Requirements**—the security, confidentiality, and integrity of the product
- 16 **Cultural Requirements**—human and sociological factors
- 17 **Legal Requirements**—conformance to applicable laws

**Project Issues**—issues relevant to the project that builds the product

- 18 **Open Issues**—as yet unresolved issues with a possible bearing on the success of the product
- 19 **Off-the-Shelf Solutions**—ready-made components that might be used instead of building something from scratch
- 20 **New Problems**—problems caused by the introduction of the new product
- 21 **Tasks**—things to be done to bring the product into production
- 22 **Migration to the New Product**—tasks to convert from existing systems
- 23 **Risks**—the risks that the project is most likely to incur
- 24 **Costs**—early estimates of the cost or effort needed to build the product
- 25 **User Documentation**—the plan for building the user instructions and documentation
- 26 **Waiting Room**—requirements that might be included in future releases of the product
- 27 **Ideas for Solutions**—design ideas that we do not want to lose

Browse through the template in Appendix A before you go too much further in this book. You will find a lot of information about writing requirements, plus much food for thought about the kinds of requirements you are looking for.

Throughout this book, we will refer to requirements by their type—that is, one of the types as shown in the template’s table of contents.

## The Snow Card

Whereas the template is a guide to *what* to write about, the snow card is a guide to *how* to write it. Individual requirements have a structure—a set of attributes, where each attribute contributes something to your understanding

---

*Any number of automated tools are available for recording, analyzing, and tracing requirements.*

---

of the requirement, and to the precision of the requirement, and thereby to the accuracy of the product's development.

Before we go any further, we must point out that although we call this device a card, and we use cards in our courses, and this book is sprinkled with diagrams featuring this card, we are not advocating writing all your requirements on cards. Some good things can be realized by using cards when interviewing stakeholders and quickly scribbling requirements as they come to light. Later, these requirements are recorded in some electronic form; at that time, their component information is filled in. Thus any reference to "card" should be taken to mean (probably) a computerized version.

At first glance, the card might seem rather bureaucratic. (See Figure 2.9.) We are not seeking to add to your requirements burden, but rather to provide a way of accurately and conveniently gathering the needed information—each of the attributes of the snow card makes a contribution. We shall explain these as we work our way through this book.

**Figure 2.9**

The requirements shell or snow card, consisting of a 5-inch by 8-inch card, printed with the requirement's attributes, that is used for our initial requirements gathering. Each of the attributes contributes to the understanding and testability of the requirement. Although a copyright notice appears on the card, we have no objections to any reader making use of it for his or her requirements work, provided the source is acknowledged.

The diagram shows a rectangular card with the following fields and labels:

- Requirement #:** Unique id (Label: *The type from the template*)
- Requirement Type:** (Label: *The type from the template*)
- Event/BUC/PUC #:** (Label: *List of events / use cases that need this requirement*)
- Description:** A one sentence statement of the intention of the requirement
- Rationale:** A justification of the requirement
- Originator:** The person who raised this requirement
- Fit Criterion:** A measurement of the requirement such that it is possible to test if the solution matches the original requirement
- Customer Satisfaction:** (Label: *Degree of stakeholder happiness if this requirement is successfully implemented. Scale from 1 = uninterested to 5 = extremely pleased.*)
- Customer Dissatisfaction:** (Label: *Measure of stakeholder unhappiness if this requirement is not part of the final product. Scale from 1 = hardly matters to 5 = extremely displeased.*)
- Priority:** A rating of the customer value
- Conflicts:** (Label: *Other requirements that cannot be implemented if this one is*)
- Supporting Materials:** (Label: *Pointer to documents that illustrate and explain this requirement*)
- History:** Creation, changes, deletions, etc.

**Volere**  
Copyright © Atlantic Systems Guild

## Your Own Requirements Process

The itinerant peddler of quack potions, Doctor Dulcamara, sings the praises of his elixir—it is guaranteed to cure toothache, make you potent, eliminate wrinkles and give you smooth beautiful skin, destroy mice and bugs, and make the object of your affections fall in love with you. This rather fanciful libretto from Donizetti's opera *L'elisir d'amore* points out something that, although very obvious, is often disregarded: There is no such thing as the universal cure.

We really would like to be able to present you with a requirements process that has all the attributes of Doctor Dulcamara's elixir—a process that suits all projects for all applications in all organizations. We can't. We know from experience that every project has different needs. However, we also know that some fundamental principles hold good for any project. So instead of attempting to provide you with a one-size-fits-all magic potion, we have distilled our experiences from a wide variety of projects to provide you with a set of foundation activities and deliverables that apply to any project.

The process described in this book is made up of the things you have to do to successfully discover the requirements. Likewise, the deliverables presented here are the foundation for any kind of requirements activity. Our intention is not to say that there is only one true path to requirements Nirvana, but rather to give you the components you need for successful requirements projects.

As you read this book, think about how you can use these components within the constraints of your own culture, your own environment, your own organizational structure, and your own chosen way of product development.

To adapt this process, you should understand the deliverables it produces—the rest of this book will discuss these items in detail. Once you understand the content and purpose of each deliverable, ask how each one (provided it is relevant) would best be produced within your project environment using your resources:


- What is the deliverable called within your environment? Use the definitions of the terms used in the generic process model and identify the equivalent deliverable in your organization.
- Is this deliverable relevant for this project?
- How much do you already know about this deliverable? Do you know enough to be able to avoid devoting additional time to it?
- Who produces the deliverable? Understand which parts of the deliverable are produced by whom. Also, when several people are involved, you need to define the interfaces between them.
- When is the deliverable produced? Map your project phases to the generic process.

---

*We have distilled experience from a wide variety of projects to provide you with a set of foundation activities and deliverables that apply to any project.*

---

### READING



Brooks, Fred. *No Silver Bullet: Essence and Accidents of Software Engineering*, and "No Silver Bullet Refired." *The Mythical Man-Month: Essays on Software Engineering*, twentieth anniversary edition. Addison-Wesley, 1995. This is possibly the most influential book on software development; it certainly is timeless.

- Where is the deliverable produced? A generic deliverable is often the result of fragments that are produced in a number of geographical locations. Define the interfaces between the different locations and specify how they will work.
- Who needs to review the deliverable? Look for existing cultural checkpoints within your organization. Do you have recognized stages or phases in your projects at which peers, users, or managers must review your specification?

The generic model describes deliverables and procedures for producing them; our intention is that you decide how you use them.

We also point you to Chapter 9 of this book, entitled *Strategies for Today's Business Analyst*. This chapter considers how you might approach your requirements projects. We suggest that before you become too involved in the mechanics of requirements discovery, you think about the strategy that is most suitable for you.

## Formality Guide

There is every reason to make your requirements discovery and communication as informal as possible. We say “as possible” because it is not so much what you would like as what your situation demands—often the degree of formality will be dictated by factors beyond your control. For example, you may be developing software using contracted outsourced development. In this case, there is a clear need for a complete written requirements specification. In other cases, the way you communicate your requirements can be informal to the point that a portion of the requirements are not written, or partially written, and communicated verbally.

We have included a formality guide to suggest where you might take a more relaxed approach to recording requirements, as well as those times when you should rightly be more systematic with your requirements discovery and communication. These are the conventions you will encounter as you move through this book.



**Rabbit—small, fast, and short-lived.** Rabbit projects are typically smaller projects with shorter lifetimes, where close stakeholder participation is possible. Rabbit projects usually include a lesser number of stakeholders.

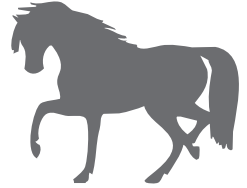
Rabbit projects are usually iterative. They discover requirements in small units (probably one business use case at a time) and then implement a small increment to the working functionality, using whatever has been implemented to solicit feedback from the stakeholders.

Rabbit projects do not spend a great deal of time writing the requirements, but use conversations with the stakeholders as a way to elaborate

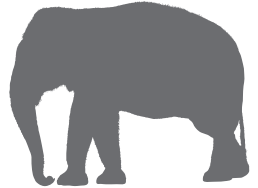
the requirements written on story cards. Rabbit projects almost always co-locate the business knowledge stakeholders with the business analysts and the developers.

**Horse—fast, strong, and dependable.** Horse projects are probably the most common corporate projects—they are the “halfway house” of formality. Horse projects need some formality—it is likely that there is a need for written requirements so that they can be handed from one department to another. Horse projects have medium longevity and involve more than a dozen stakeholders, often in several locations, factors that necessitate consistently written documentation.

If you cannot categorize your own project, think of it as a horse.



**Elephant—solid, strong, long life, and a long memory.** An elephant project has a need for a complete requirements specification. If you are outsourcing the work, or if your organizational structure requires complete, written specifications, you’re an elephant. In certain industries, such as pharmaceuticals, aircraft manufacture, or the military, regulators demand not only that full specifications be produced, but also that the process used to produce them be documented and auditable. Elephant projects typically have a long duration, and they involve many stakeholders in distributed locations. There are also a large number of developers, necessitating more formal ways of communicating.



## The Rest of This Book

We have described—briefly—a process for discovering, communicating, and verifying requirements. The remainder of this book describes the various activities in this process, along with their deliverables, in some detail. Feel free to jump to any chapter that is of immediate concern—we wrote the chapters in more or less the order in which you would do each of the activities, but you don’t have to read them that way.

And please, while you are reading this book, be constantly asking yourself how you will do the things we describe. After all, it is you who has to do them.

We hope find useful ideas, processes and artifacts, in the rest of this book. We also hope you enjoy reading and using it.

*This page intentionally left blank*



# Scoping the Business Problem

# 3

*in which we establish a definition of the business area to be changed, thereby ensuring that the project team has a clear vision of what their project is meant to achieve*



Let's revisit our fundamental premise: If a piece of software (or any device or service) is to be built, it must be **optimally valuable to its owner**.

From this statement, you can safely infer that if you are to know what is optimally valuable, **you must firstly determine what the owner is actually doing, who he is doing it with or for, and why he wants to do it**. To put that another way: What are **the scope, stakeholders**, and goals? Hold that thought—we will come back to it in a moment.

In the Volere process model, as shown in Figure 3.1, **the first activity is the Project Blastoff**. It is during this activity—typically a **meeting of the key stakeholders**—that you decide the **key factors** that, taken together, **determine the viability of the requirements project**.

Te juntas con los principales stakeholders y determinas factores claves que determinaran la viabilidad del proyecto.

*If a piece of software is to be built, it must be optimally valuable to its owner.*

## Project Blastoff

The blastoff identifies **the boundary of the work area** the product is to become a part of, and determines the **purpose the product is to fulfill**. It also **identifies the stakeholders—those people who have an interest in the success of the product**. Other deliverables from the blastoff **qualify the project**, and are used as inputs to **subsequent requirements discovery activities**.

Identifican límite y proposito

You might know this activity as “project kick-off,” “project initiation,” “project launch,” or any one of the many other names given to it. Whatever you call it, its purpose is to **lay the groundwork so your requirements discovery is efficient and effective**.

Determinan si los req son eficientes y efectivos

**The blastoff produces a number of deliverables**. You might be given some of these by your project manager; others, you might have to dig out for yourself. In any event, you need these deliverables if you are to have a successful

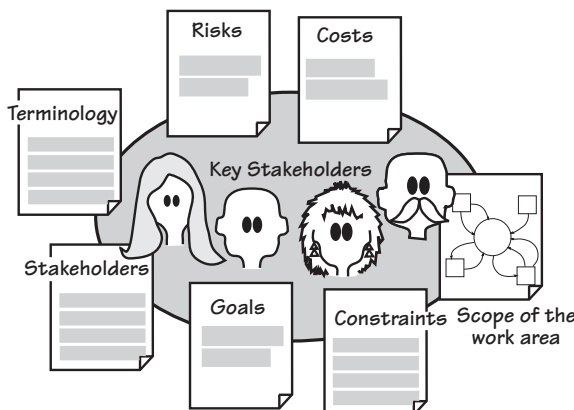


- **The stakeholders:** The people with an interest in the product. This group includes anyone who has some influence on the outcome or has knowledge needed to uncover the requirements for the product.
- **Constraints:** Restrictions on the scope or style of the product. These items include predetermined design solutions that must be used, constraints on changing the way that business processes are currently implemented, and the time and money that are available for the project.
- **Names:** Any special terminology to be used in this project.
- **Relevant facts and assumptions:** Are there any special facts people need to know? Or are there assumptions being made that may affect the outcome of the project?
- **The estimated cost:** Some of the deliverables from the blastoff provide input to the estimating process and allow fairly good estimates to be made early in the project. This is not really a requirements issue, but as the requirements deliverables are prime input, your project management will thank you for it.
- **The risks:** Possibly a short risk analysis to reveal the main risks faced by the project. Someone skilled at risk assessment should perform this analysis.

Taken together (see Figure 3.2), these deliverables give you enough information to make the final deliverable from the blastoff:

- **Go/no go decision:** Is the project viable, and does the cost of producing the product make it worthwhile? Do you have enough information to proceed with the requirements activity, or should you ask for extra time to gather more information and build a better foundation?

Hay que considerar los "materiales" que se tienen antes de empezar la "obra", el proyecto, esto incluye lo que se sabe o no de tecnologías



**Figure 3.2**

The blastoff activity assembles enough information to ensure that the project can proceed smoothly. It also verifies that the project is viable and worthwhile. Most of the outputs serve as the foundation for the trawling activity about to come; the risks and costs are used by project management.

## Formality Guide

The blastoff deliverables—especially scope, stakeholders, and goals—are needed by any project, regardless of its size or aspirations for informality. Even a small change to an existing system needs to ask these questions. Any project must have a clear understanding of its goals if it is to avoid wandering aimlessly. Additionally, a project must understand the work to be improved; otherwise, it runs the risk of producing a solution in search of a problem.

In the previous chapter we spoke about rabbit, horse, and elephant projects. These types relate to formality and their differences are as follows.

**Rabbit** projects should have a sketch of the work scope diagram pinned to the wall close to their user stories; close by should be a list of stakeholders. Finally, written with a broad marker pen on the wall, are the project goals. Rabbits will probably have only a brief blastoff meeting at best, with most of the consensus on the blastoff coming from wikis, phone calls, and other informal interactive means. Despite the relative informality, we cannot stress how important it is to document the work scope, and to ensure that you are thinking about the *work*, and not just the intended product.

Horse projects should be more formal and hold a blastoff meeting; they should also record their deliverables and communicate them to the appropriate stakeholders. Horse projects might benefit from producing a first-cut sketch of the guessed-at product to ensure all stakeholders (we are thinking of the nontechnical ones) understand where the project is headed. This sketch must be destroyed as soon as the stakeholders have confirmed it.

Elephants have a lot to lose by not having the blastoff deliverables firmly in place before proceeding further in the product development process. In most cases, the deliverables are discovered during meetings with the key stakeholders, and the results are recorded and distributed. Elephants should take the additional step of having the quality assurance (QA) people test their blastoff deliverables: Elephant projects are critical, and costly if they make errors, so the foundation of the requirements must be proved to be rock solid. Risk analysis and cost estimation are important to elephant projects; having a clearly defined and properly understood work scope is crucial.

The degree of formality you apply to the blastoff deliverables varies, but that does not stretch to ignoring them. This chapter explains these deliverables.

## Setting the Scope

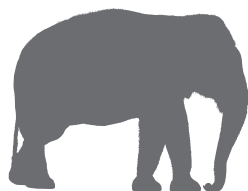
Let's go back to optimal value and the owner.

If the owner is an organization, you have to understand the business processes and aims of that organization, or, as is the case for most projects, the

Tener cosas en la pared,  
pero claras



Puede server el sketch pero  
luego dejar de utilizarlos,  
solo para modo explicación



Se distribuyen los  
resultados de las  
reuniones y tenes gente q  
testea esos resultados  
(los 7 variables q salen de  
la reunión)

part of the organization your project will affect. If the owner is an individual using some personal computer software or a mobile device app, you still have to understand what that owner is doing, and what he wants to do, if you are to deliver optimal value. In any event, if you want to build something valuable, then you have to understand what the owner values and what he is trying to achieve when he uses your product.

It is unlikely that you will ever have to study the entirety of the owner's business. Almost certainly, you need to examine only part of the business—the part that you will change when you install the product you intend to build. Let's call this part of the business "the work"; the first task in the product development life cycle is to define the precise scope of that work. You need to know which parts of the business are included in the work, and which parts can be safely excluded.

Figure 3.3 illustrates the relationship between different scopes: the scope of the work, the scope of the entire organization within the outside world, and the potential scopes of the product your project is to build.

For the moment, you are intentionally ignoring any proffered solution; until you understand what the solution is to be used for, there is little point in spending time on it. Instead, you should step back and look at the work that the owner values and, most importantly, define its scope.

The scope you are interested in at the beginning of the requirements project is the scope of the owner's work—specifically, the piece of work that the owner would like to change and improve. This work might be a commercial business activity, some scientific or technical work, or a game—anything at

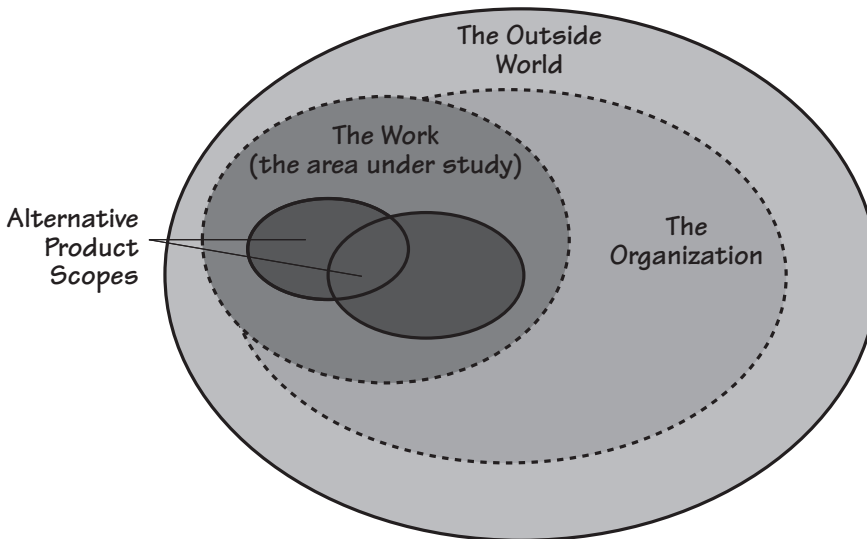
Entender el área de trabajo para hacer un bien trabajo

No es lo mismo el scope de tu proyecto que el de toda la organización

---

*Step back and look at the work that the owner values, and define its scope.*

---



**Figure 3.3**

The work is the part of the organization that you need to study to discover the requirements. The work is usually connected to other parts of the organization and to the outside world. You must study the work well enough to understand how it functions. This understanding will enable you to come up with alternative scopes for the product and eventually choose the one to build.

all that involves some kind of **meaningful activity**. The work might be currently automated, it might be manual, or it might take advantage of a combination of these approaches that makes use of several different devices. At this stage it makes no difference whether your product is to be sold to an outside customer or used within your own organization. As long as it involves some processing and some data, we shall refer to it as “work,” and you must know the scope of it.

### Separate the Work from its Environment

**Any piece of work must be connected to at least one other piece of work;** there can be no such thing as a piece of work without any external connections. If there were no connections, then the work would be useless; it would not have any outputs. It is also true that any activity within that work is connected to at least one other activity. With that point in mind, you can separate the activities that you are interested in—the ones inside your work—from other activities that hold no interest for you. The latter activities are considered to be outside the scope of your work.

To make this separation, you rely on the simple fact that **all activities are driven by data**. We spoke a moment ago about activities being connected to other activities, with this **connection being a flow of data**. That is to say, an activity produces some data and then hands this data on to another activity. When the next activity in turn receives its incoming flow of data, it is triggered to do whatever processing it is meant to do, and produces a different data output, which is in turn sent to another activity. Thus these flows of data are the connections between activities.

**By identifying these data connections, you determine the boundary of your interest.** To do so, **you say that you include in your work area the activity that produces a certain packet of data, but you are not interested in the activity that receives this data packet.** By drawing a line to represent your work boundary between similarly coupled activities, **you create a partition that eventually includes all the activities that are to be part of your work.**

You cannot effectively describe the work area by simply looking at the processors that do the work. The problem is that almost all processors—computers, humans, and mechanical devices—are capable of carrying out more than one type of activity. Nor can you effectively describe the work area using words alone: It is difficult, if not impossible, to accurately describe what to study and what to ignore.

**To achieve the optimal value for the owner, study enough of the owner's work to identify what is valuable.** Then decide **how much effort can be expended to build a product to improve the valuable part of the work.** Clearly, **if you study too much, then the project will produce less value** because you are wasting resources studying unnecessary activities. Conversely, **if you study too little of the owner's work, you may fail** to achieve

---

*To achieve the optimal value for the owner, study enough of the owner's work to identify what is valuable.*

---

La aplicación que  
hagas tiene que estar  
conectada con otra  
parte y esa  
comunicación es data

the optimal value because you do not know enough about the business to make the correct value decisions.

Let's look at an example of how you can isolate the activities to be studied from those you will ignore. We said earlier that activities have data flowing between them. This simple fact enables you to construct a diagram that shows the data that flows from a non-interesting activity to the collection of activities that you find interesting. A sample of this diagram—called a *context diagram* or a *work scope diagram*—appears in Figure 3.5. Before examining it, however, you need a little background on this diagram.

## IceBreaker

IceBreaker is a case study we have put together to illustrate the requirements process. IceBreaker uses data from the environment to predict when ice will form on roads. It then schedules trucks to treat the roads with de-icing material (a salt compound) before the roads can become dangerous. The IceBreaker case study uses subject-matter knowledge from the many ice forecasting and road de-icing systems, and other products produced by Vaisala (U.K.) Limited and Vaisala Worldwide. We acknowledge Vaisala's permission to use its material and the company's kind cooperation. Figure 3.4 depicts a weather station used by IceBreaker.

Imagine IceBreaker is your project. You work for Saltworks Systems, and you are responsible for producing the requirements specification. The first customer for the IceBreaker product is the Northumberland County



**Figure 3.4**

This weather station transmits data about the weather and road surface conditions to IceBreaker, which uses the information to predict ice formation. These predictions are used to dispatch trucks to treat the roads with de-icing material. Photo of Vaisala Weather Station ROSA courtesy of Vaisala. [www.vaisala.com](http://www.vaisala.com).



Highways Department. Northumberland is a county in the northeast of England, tucked up under the border with Scotland, with serious snow and icy conditions in winter. The Highways Department is responsible for keeping the roads free of ice that is likely to cause accidents; it has agreed to provide expertise and information for you to build the optimally valuable product for the department.

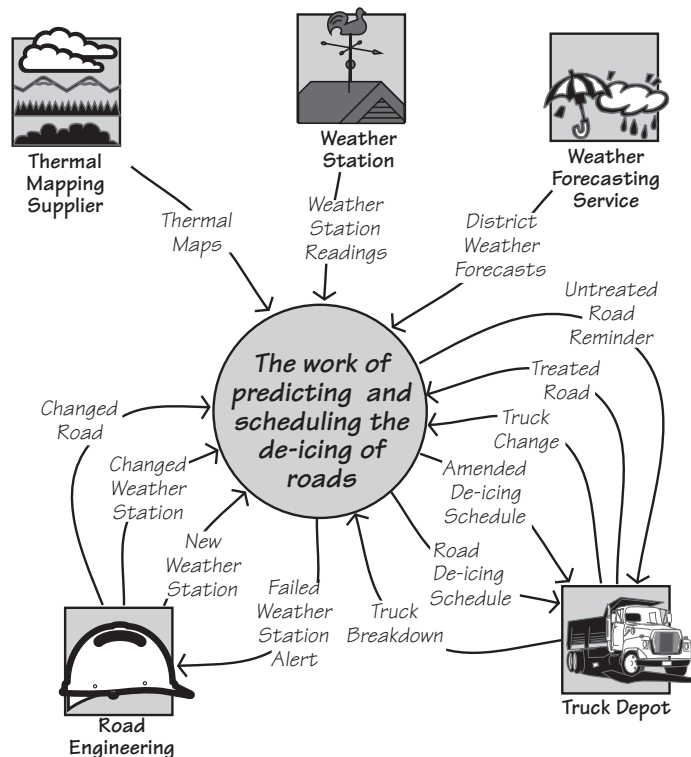
### First-Cut Work Context

A lot of activities are carried out as part of the process in which the IceBreaker work predicts whether there is a need to treat a road: A weather forecast is generated, the temperature at the surface of the road is taken, predictions about which roads will freeze are made, trucks are dispatched, and so on. **To deliver optimal value, you must determine which of these activities should be studied because they have the possibility for improvement, and which can be safely excluded from the study.**

To illustrate your decisions, collect **all the activities to be studied and hide them (for the moment) in the central circle.** Put the connected activities that are not part of your study outside the circle, and show the data that connects them. Using this approach, our first draft of the context diagram for the IceBreaker work is shown in Figure 3.5.

Figure 3.5

The work context diagram identifies the scope of the work to be studied. It shows the work as a single activity, surrounded by the adjacent systems. The named arrows represent the data that flows between the work and the adjacent systems. The adjacent systems are the activities that you have decided not to study.



The context diagram shows the work to be studied, as well as those activities you decide not to study. The latter activities are called *adjacent systems*. The purpose of the context diagram is to show the processing responsibilities of the work and the responsibilities of the adjacent systems. But be aware that the responsibilities are in reality defined by the flows of data on the context diagram. As an example, the flow called *Truck Change* advises the work about changes to the de-icing trucks—for example, new trucks that have been added to the fleet, trucks that have been taken out of service, and modifications to trucks that would affect the way that they are scheduled. Why is this flow there? The work needs this information when it allocates a truck to each road that needs treatment as part of the production of the de-icing schedule. But what if you changed this responsibility? What if the depot became responsible for allocating trucks to the icy roads? In that scenario, the flow would be different. In fact, the *Truck Change* flow would not appear on the work context diagram at all, as the activities that are triggered by this data flow have become the responsibility of the adjacent system.

This leads to us saying that the data flows around the boundary of the work are the clearest indication of its processing capabilities. By defining these flows, you define the precise point at which the processing of the work ends and that of the adjacent system starts, and vice versa.

One problem commonly arises in setting the context: Often we see product-centric contexts that contain only the intended software product. Remember that you are investigating some work, and the eventual product will become *part* of that work. To specify the most valuable product, you must understand the work into which that product is to be deployed. In most cases, projects that restrict their study to only what they think the product will contain build less useful products; in fact, they often omit functionality that would have been valuable to the owner. As a rule of thumb for commercial projects, if you do not have any humans inside your work context, then chances are that your work context is too narrow.

Also consider the possibility that by enlarging the scope of the work, you find other potential areas for automation or other types of improvements that could be valuable. All too often, before we understand the work, we think of an automation boundary, and we never rethink it. Of course, then the “hard stuff”—the work that we did not intend to automate—is not considered. By casting your net wider, you usually find aspects of the work that would benefit from automation or some other improvement, and in the end turn out to be cheaper than first thought. The moral of the story: First understand the work, and then decide which product is most valuable to that work.

---

*The work context shows where the responsibilities of the work and the responsibilities of the adjacent systems start and end.*

---

Separa las responsabilidades de tu proyecto con las de los sistemas adyacentes

No entendí pq lo subrayado

---

*The moral of the story: First understand the work, then decide which product provides the best value to that work.*

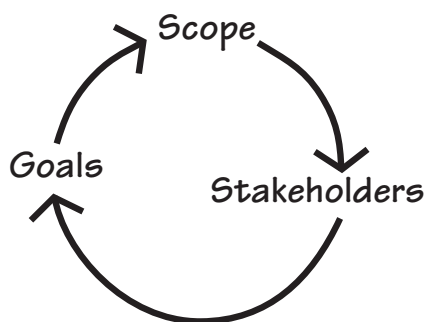
---

## Scope, Stakeholders, and Goals

Scope is not all there is to getting the requirements project off the ground. To build the right product, you have to understand the extent of the work;

**Figure 3.6**

The scope, stakeholders, and goals are not decided in isolation from one another. Rather, the scope of the work indicates the stakeholders who have an interest in the work; the stakeholders, in turn, decide what they want the goals of the project to be.



the people who do it, influence it, or know about it; and the outcome that those people are trying to achieve. This is the **trinity of scope, stakeholders, and goals**, as shown in Figure 3.6.

The scope is the extent of the business area affected by the product. Because it is defining a part of a real-life organization, the scope points to the **stakeholders**—the people who have an interest in, or an effect on, the success of the work. The stakeholders, in turn, decide the **goals**, which is the improvement the business wants to experience when the product is installed.

There is **no particular order** to deciding what these factors should be. Most projects start with scope, but it is not obligatory—you use whatever information is to hand first. You have to iterate between the three factors until you have stabilized them, but this is almost always a short process when your organization knows why it wants to invest in the project.

## Stakeholders

---

*Stakeholders are the source of requirements.*

---

The next part of the trinity is the stakeholders. Stakeholders include anyone with an interest in, or an effect on, the outcome of the product. The owner is the most obvious stakeholder, but there are others. For example, the intended users of the product are stakeholders—they have an interest in having a product that does their work correctly. A subject-matter expert is an obvious stakeholder; a security expert is a less obvious stakeholder but one who must be considered for any product that could hold confidential or financial information. Potentially dozens of stakeholders exist for any project. Remember that you are trying to establish the optimal value for the owner, and that probably means talking to many people, all of them are potentially sources of requirements.

The stakeholder map (Ian Alexander calls it an *onion diagram*) in Figure 3.7 identifies common classes of stakeholders that might be represented by one or more roles in your project. Let's look a little more closely at this map.

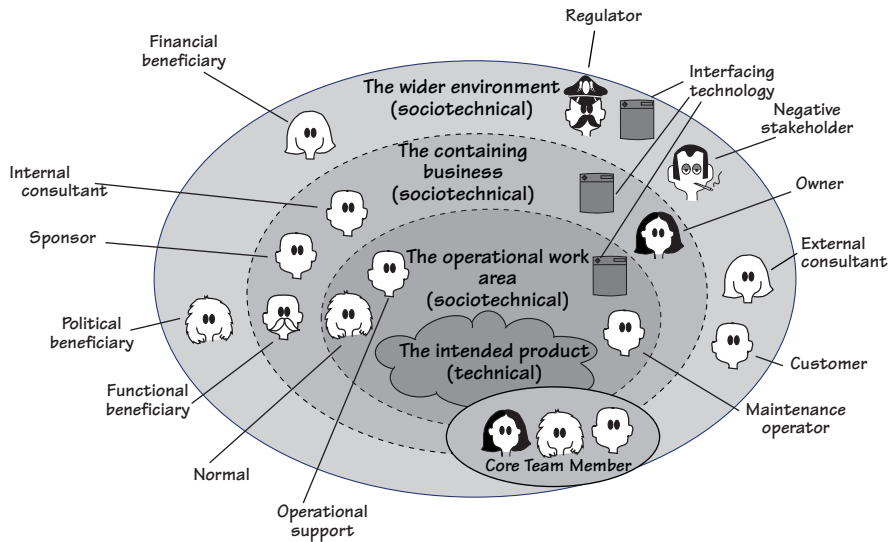


Figure 3.7

This stakeholder map shows the organizational rings surrounding the eventual product, and the classes of stakeholders who inhabit these rings. Use this map to help determine which classes of stakeholders are relevant to your project and which roles you need to represent them.

At the center of the stakeholder map is the *intended product*. Notice that it has a vague, cloud-like shape—this is intentional. At the start of the requirements activities you cannot be certain of the precise boundaries of the product, so for the moment we will leave it loosely defined. Surrounding the intended product is a ring representing the *operational work area*—stakeholders who will have some direct contact with the product inhabit this space. In the next ring, the *containing business*, you find stakeholders who benefit from the product in some way, even though they are not in the operational area. Finally, the outer ring, the *wider environment*, contains other stakeholders who have an influence on or an interest in the product. Note that the detailed and multiple involvement of the *core team members*—analysts, designers, project manager, and so on—is emphasized by the fact they span all the rings.

Because so many classes of stakeholders exist, it is helpful to discuss some of the more important ones. After we have discussed these stakeholders, we will point you at a way of formalizing them with a stakeholder analysis template.

## The Sponsor

We have said—often, and it shall be repeated—that the product has to provide the optimal value to its owner. However, for many products you do not, and usually cannot, have direct access to the owner. Many projects are carried out by commercial organizations, which are strictly speaking owned by their shareholders. Naturally enough, you can't go and talk to all the shareholders, nor are you likely to get access to the board of directors. In this case,

## READING

For more on stakeholder analysis, refer to Alexander, Ian, Neil Maiden, et al. *Scenarios, Stories, Use Cases Through the Systems Development Life-Cycle*. John Wiley & Sons, 2004.

---

*The sponsor pays for the development of the product.*

---

the usual course of action is to appoint a sponsor for the project to represent the owner's interest; in many cases, the resources for the development of the new product that come from a sponsor's budget. You will find that the sponsor is concerned with project issues, and will be instrumental in setting some of the constraint requirements. We will deal with constraints in a little while.

On the simple basis that "money talks," the sponsor, by paying for the development, has the final say in what that product does, how it does it, and how elaborate or how sparse it must be. In other words, the sponsor is the ultimate arbiter of which product will yield the optimal value.

You cannot proceed without a sponsor. If no one is representing the interest of the organization at large, then there is little point in proceeding with the project. The sponsor is most likely to be present at the blastoff meeting (you should be worried if the sponsor is not there) and is most likely one of these people:

- *User Management:* If you are building a product for in-house consumption, the most realistic sponsor is the manager of the users who will ultimately operate the product. Their department, or its work, is the beneficiary of the product, so it is reasonable that the cost of construction is borne by the departmental manager.
- *Marketing Department:* If you build products for sale to people outside your organization, the marketing department may assume the role of sponsor and represent the eventual owners of the product.
- *Product Development:* If you build software for sale, the budget for its development might be with your product manager or strategic program manager, in which case one of those would be the sponsor.

Consider your own organization, its structure and job responsibilities: Which people best represent the owner? Who pays for product development? Who, or what, reaps the benefit of the business advantage that the product brings? Whose values do you have to consider when you are determining what the end product is to do?

Do whatever you need to do to find your sponsor; your project will not succeed without one.

Let's assume the sponsor (in this case an owner representative) for the Ice-Breaker project is Mack Andrews, the CEO of Saltworks Systems. Mr. Andrews has made the commitment to invest in building the product. You record this agreement in Section 2 of your requirements specification:

*The sponsor of the project is Mack Andrews, the CEO of Saltworks Systems. He has said that it is his goal to develop this product to appeal to a broader range of markets in other countries, including airports and their runways.*

There are several things to note here. First, you name the sponsor. It is now clear to everyone on the project that Mack Andrews takes responsibility for investing in the product, and so will be the final arbiter on scope changes. Second, other information is provided about the sponsor that will be used as the project proceeds and may have a bearing on some of the requirements—particularly, the usability, adaptability, and “productization” requirements.

For more details on usability, adaptability, productization, and other types of requirements, refer to the template in Appendix A.

## The Customer

The customer buys the product once it is developed, becoming the product’s new owner. To persuade the customer to purchase the product, you have to build something that your customer will find valuable and useful and pleasurable.

Perhaps you already know the names of your customers, or perhaps they are hundreds or thousands of unknown people who might be persuaded to pay for your product. In either case, you have to understand them well enough to understand what they find valuable and what they will buy.

There is a difference between customers who buy for their own use and those who buy for use by others. When the product is a retail product and the customer and the owner are the same person, then that person’s own values are of supreme importance to you. Is the customer looking for convenience? Most people are, and if so, you have to discover what the customer is doing that you can make more convenient. How much convenience will he find valuable?

When customers buy for use by others, the owner is presumably an organizational customer. Your interest lies in what the organization is doing, and what it considers valuable. That is, what can your product do that will make the users within the organization more productive, efficient, quicker, or whatever other quality it is seeking?

Even if you are developing open-source software, you still have a customer; the difference is simply that no money changes hands.

You must understand what appeals to your customers, and what they value. What will they find useful? What will they pay for? Understanding your customer correctly makes a huge difference to the success of your product.

For the IceBreaker product, the Northumberland County Highways Department has agreed to be the first customer.<sup>1</sup>

*The customer for the product is the Northumberland County Highways Department, represented by director Jane Shaftoe.*

As there is a single customer (at this stage), it would, of course, be advisable to invite her to participate as a stakeholder in the project. This kind of outreach results in the customer being actively involved in selecting

---

*The customer buys the product. You have to know this person well enough to understand what he finds valuable and, therefore, what he will buy.*

---

### FOOTNOTE 1

The original Vaisala de-icing prediction system was built for the Cheshire County Council. The designers of the product were Thermal Mapping International and The Computer Department. The product is now installed by all counties in the United Kingdom and has thousands of customers overseas in most cold-weather countries. [www.vaisala.com](http://www.vaisala.com).

which requirements are useful, choosing between conflicting requirements, and making the requirements analysts aware of her values, problems, and aspirations.

Saltworks Systems has further ambitions for the IceBreaker product. In the earlier statement about the sponsor, he said that he wanted an ice forecasting system that could be sold to road authorities and airports in other counties and other countries. If you plan to build the product with this aim, then your requirements specification should include an additional customer statement:

*Potential customers for the product include all counties in the United Kingdom, northern parts of North America, and northern Europe and Scandinavia. A summary of the requirements specification will be presented to the Highways Department managers of selected counties, states, and countries for the purpose of discovering additional requirements.*

It is clear that the customer must always be represented on the project. Where many potential customers exist, you must find a way of representing them in your project. This representation can come from the marketing department, senior users from one or more of your key customers, or a combination of domain and usability experts from within your organization. We will also later discuss personas as a way of representing the customers. The nature of your product, the structure of your organization, your customer base, and probably several other factors decide which roles within your organization are able to represent the customers.

### Users: Understand Them

We are using the term *users* to mean the people who will ultimately be the hands-on operators of your product. The stakeholder map (Figure 3.7) refers to them as *normal operators*, *operational support*, and *maintenance operators*. For in-house products, the users are typically the people who work for the project sponsor. For personal computer or mobile device products, the user and the owner are often the same person.

Identifying your users is the first step in understanding the work they do—after all, your product is intended to improve this work. Additionally, you have to learn what kind of people they are so you provide the right user experience (we look at this issue later when we discuss usability requirements). You have to bring about a product that your users are both able to use and want to use. Obviously, the better your understanding of your users, the better the chance you have of specifying a suitable product for them.

---

*The purpose of identifying the users is to understand what they are doing and which improvements they consider to be valuable.*

---



Different users make different demands on your product. For example, an airline pilot has very different usability requirements from, say, a commuter buying a ticket on a rail system. If your user were a commuter, then a “person without cash” and a “person with only one arm free” would raise their own usability requirements.

When developing consumer products, mass-market software, or websites, you should consider using a *persona* as the user. A persona is a virtual person who is archetypical of most of your users. By determining the characteristics of this persona to a sufficient degree, the requirements team can know the correct requirements to satisfy each of the personas. You should decide at this stage if you will use personas, but they can be more fully developed later. More information is provided on personas in Chapter 5, Investigating the Work.

The consideration of potential users is vital for agile development. Too many teams operate with only one user asked to supply the requirements for a product, and little or no consideration given to what will happen when the product is released to a wider audience. We strongly urge you to always consider the broadest spectrum of users and, at the very least, to choose user stakeholders from both ends of that spectrum.

Having identified your users, you have to record them. For example, in the Icebreaker project we had users in the following categories:

- Qualified Road Engineers
- Clerks in the truck department
- Managers

For each category of user, write a section in your specification to describe, as fully as time allows, the attributes of your users. Consider these possibilities:

- Subject-matter experience: How much help do they need?
- Technological experience: Can they operate the product? Which technical terms should be used?
- Intellectual abilities: Should tasks be made simpler? Or broken down to a lower level?
- Attitude toward the job: What are the users' aspirations?
- Education: What can you expect your user to know?
- Linguistic skills: Not all your users will speak or read the home language.
- And most importantly, what is it about their work that they most wish to improve?

Also, for each category of user, identify the particular attributes that your product must cater to:

## READING

Don Gause and Jerry Weinberg give a wonderful example of brainstorming lists of users in their book: Gause, Don, and Gerald Weinberg. *Exploring Requirements: Quality Before Design*. Dorset House, 1989. Although this text is getting a little long in the tooth, its advice is still relevant. Most of Jerry Weinberg's books are available as Kindle books or at Smashwords.

---

***A persona is a virtual person that is archetypical of most of your users.***

---

- People with disabilities: Consider all disabilities. This, in some cases, is a legal requirement.
- Nonreaders: Consider people who cannot read and people who do not speak the home language.
- People who need reading glasses: This is particularly near and dear to one of the authors.
- People who cannot resist changing things like fonts, styles, and so on.
- People who will probably be carrying luggage, large parcels, or a baby.
- People who do not normally use a computer.
- People who might be angry, frustrated, under pressure, or in a hurry.

We realize that writing down all this stuff seems like a chore. However, we have found that taking the time to record it so other people can read it is one of the few ways for you to demonstrate that you understand it. The users are so important to your cause that you must understand what kind of people they are and which capabilities they have. To wave your hands and say, “They are graphic designers” or “They want to buy books on the Web,” falls short of the minimum level of understanding.

Another category of stakeholder in the operational work area is the maintenance operator. Your product is likely to have maintainability requirements, and you can learn about them from this person.

Operational support is another source of requirements relating to the operational work area. Roles that are sources of these requirements include help desk staff, trainers, installers, and coaches.

At this stage, any users you identify are *potential* users. That is, you do not yet precisely know the scope of the product—you determine this later in the requirements process—so you are identifying the people who might possibly use, maintain, and support the product. Remember that people other than the intended users (e.g., firefighters, security personnel) might end up having hands-on contact with your product. It is better to identify superfluous users than to fail to find them all because each different category of user will have some different requirements.

---

***People other than the intended users might end up having contact with your product. It is better to identify superfluous users than to fail to find them all.***

---

“Every context is composed of individuals who do or do not decide to connect the fate of a project with the fate of the small or large ambitions they represent.”

—Bruno Latour, *ARAMIS or the Love of Technology*


## Other Stakeholders

There are more people you have to talk to so that you can find all the requirements. Your contact with these people might be fleeting, but it is nevertheless necessary. Any of them may potentially have requirements for your product.

The problem that you often face in requirements projects is that you fail to discover all the stakeholders, and thus fail to discover their needs. This shortcoming may result in a string of change requests when the product is

released to an audience that is wider than at first thought. Naturally, the people who were overlooked will not be happy. Additionally, you must consider that when any new system is installed, someone gains and someone loses power: Some people find that the product brings them new capabilities, and some people are not able to do their jobs the way they used to do them. The moral of the story is clear: Find all parties who will be affected by the product, and find their requirements.

Let's consider some other stakeholders by looking at some candidate categories. You can also see most of these groups illustrated as classes on the stakeholder map shown in Figure 3.7.

 We list our stakeholders in Section 2 of the Volere Requirements Specification Template. You can find this template in Appendix A. The list acts as a checklist for finding the appropriate stakeholders.

## Consultants

Consultants—both internal to your organization and external—are people who have expertise you need. Consultants might never touch or see your product, but their knowledge becomes part of it. For example, if you are building a financial product, a security expert is one of your stakeholders. He might never see your product, but his expertise—and this is his stake in the requirements—ensures the product is secure.

## Management

Consider any category of management. These groups show up on the stakeholder map (Figure 3.7) as classes like *functional beneficiary*, *political beneficiary*, and *financial beneficiary*. Is it a strategic product? Do any managers other than those directly involved have a stake?

Product managers and program managers are obvious sources of requirements. Project managers or leaders who are responsible for the day-to-day management of the project effort likewise have contributions to make.

## Subject-Matter Experts

This constituency may include domain analysts, business consultants, business analysts, or anyone else who has some specialized knowledge of the business subject. As a consequence, these experts are a prime source of information about the work.

## Core Team

The core team is made up of the people who are part of the building effort for the product. They may include product designers, developers, testers, business analysts, systems analysts, systems architects, technical writers, database designers, and anyone else who is involved in the construction.

You can also consider the open-source community as stakeholders—they have knowledge about technology and trends in most areas of software.

You can contact these people via open-source forums. They are usually very enthusiastic and ready to share knowledge with you.

When you know the people involved, record their names. Otherwise, use this section of the template to list the skills and duties that you think are most likely needed to build the product.

### **Inspectors**

Consider auditors, government inspectors, any kind of safety inspectors, technical inspectors, or even possibly the police. It may well be necessary to build inspection capabilities into your product. If your product is subject to the Sarbanes-Oxley Act, or any of the other regulatory acts, then inspection is crucial, as are the requirements from the inspectors.

### **Market Forces**

People from the marketing department are probably the stakeholders representing the marketplace. When you are building a product for commercial sale, trends in the market are a potent source of requirements, as is a deep knowledge of the likely consumers. Note the speed at which the smart phone and tablet markets are moving (at the time of writing). Staying ahead of the curve is vital for any consumer product.

### **Legal Experts**

Each year the world is filled with more and more laws—complying with all of them is daunting but necessary. Your lawyers are the stakeholders for most of your legal requirements.

### **Negative Stakeholders**

Negative stakeholders are people who do not want the project to succeed (remember what we said previously about losing power). Although they may not be the most cooperative individuals, you would be wise to consider them. You may find that, if their requirements are different from the commonly perceived version, and you can accommodate their requirements, your opposition could well become your supporters.

You might also consider the people who threaten your product—the hackers, defrauders, and other malevolent people. You will get no cooperation from them, but you should consider how they might mistreat your product.

### **Industry Standard Setters**

Your industry may have professional bodies that expect certain codes of conduct to be followed or certain standards to be maintained by any product built within the industry or for use by the industry.

## Public Opinion

Do any user groups for your product exist? They will certainly be a major source of requirements. For any product intended for the public domain, consider polling members of the public about their opinion. They may make demands on your product that could spell the difference between acceptance and rejection.

## Government

Some products must interact with government agencies for reporting purposes, or receive information from a government agency; other products have requirements that necessitate consulting with the government. Although the government may not assign a person full-time to your project, you should nevertheless nominate the pertinent agency as a stakeholder.

## Special-Interest Groups

Consider handicapped-interest groups, environmental bodies, foreign people, old people, gender-related interests, or almost any other group that may come in contact with your product.

## Technical Experts

Technical experts do not necessarily build the product, but they will almost certainly be consulted about some part of it. For the stakeholders from this constituency, consider usability experts, security consultants, hardware people, experts in the technologies that you might use, specialists in software products, or experts from any technical field that the product could use.

## Cultural Interests

This constituency is applicable to products intended for the public domain, and especially when your product is to be sold or seen in other countries. In addition, it is always possible in these politically correct times that your product could offend someone. If there is any possibility that religious, ethnic, cultural, political, gender, or other human interests could be affected by or come into contact with your product, then you should consider representatives from these groups as stakeholders for the project.

## Adjacent Systems

The adjacent systems on your work context diagram are the systems, people, or work areas that directly interact with the work you are studying. Look at each adjacent system: Who represents its interests, or who has knowledge of it? When the adjacent system is an automated one, who is its project leader

or maintainer? If these stakeholders are not available, you might have to read the adjacent system's documentation, or its code, to discover whether it has any special demands for interacting with your product. For each adjacent system you need to find at least one stakeholder.

## Finding the Stakeholders

At scoping time, you normally inspect your context model and hold a brainstorming session to identify all possible stakeholders. You do not have to start from scratch; we have constructed a spreadsheet with many categories of stakeholders, along with the kind of knowledge you need to obtain from each person. This spreadsheet (see Appendix B) cross-references the stakeholder map (Figure 3.7) and provides a detailed specification of your project's sociology. Once you have identified the stakeholder, add that person's name to the list. The complete spreadsheet is available as a free download at [www.volere.co.uk](http://www.volere.co.uk).

---

*The greatest problem concerning stakeholders is the requirements you miss when you don't find all of the stakeholders.*

---

See the Stakeholder Management Template in Appendix B, also available as a downloadable Excel spreadsheet at [www.volere.co.uk](http://www.volere.co.uk).

You will be talking to the stakeholders, so at this stage it pays to explain to them why they are stakeholders and why you need to consult them about requirements for the product. Explain specifically why their input will make a difference to the eventual product. It is polite to inform stakeholders of the amount of their time you require and the type of participation that you have in mind; a little warning always helps them to think about their requirements for the product. The greatest problem concerning stakeholders is the requirements that you miss if you do not find all of the stakeholders, or if you exclude stakeholders from the requirements-gathering process.

## Goals: What Do You Want to Achieve?

When you are busy working with your stakeholders on detailed requirements, it is very easy to go *off piste* and either spend time on irrelevancies or miss important requirements.

Your sponsor is making an investment in a project to build a product; to understand the reason behind this investment, you need to determine the precise benefits the project is to deliver. You also need a guide to help you steer your efforts toward those requirements that will make the greatest contributions to the expected business advantage.

In other words, you need to know the goal of the project. You can think of the project goal as the highest-level requirement; all the detailed requirements you collect along the way must make a positive contribution toward that goal.

Spending a little time during the blastoff to ensure that a consensus on the goal of the project has been reached will pay handsome dividends over

---

*The project goal is the highest-level requirement.*

---

the course of the project. The goal should be written clearly, unambiguously, and in a measurable way so it quantifies the benefits of the project; this quantification makes the goal testable.

How do you make a clear statement of the goal? Start with a statement of the user problem or background to the project. (We make this problem statement the first part of all our specifications; see the template in Appendix A for a suggested format.) Those stakeholders who represent the user or business side of the organization should confirm that you do, indeed, understand the problem, and that your problem statement is a fair and accurate one.

For the IceBreaker project, the business has given you this background:

“Roads freeze in winter, and icy conditions cause road accidents that may potentially kill people. Predictions at the moment rely largely on guesswork, experience, and phoned-in reports from motorists and the police. Trucks do not always get to the icy roads on time to prevent accidents, or they may arrive far too early, which results in the de-icing material being dispersed by the time the road freezes. Road treatment is sometimes indiscriminate, and this wastes de-icing material and causes environmental damage.”

You and your blastoff group must learn and articulate the business problem. Only when that is done can you discover the requirements that make the greatest contribution toward solving the problem.

Once you have a clear understanding of the problem, it is time to move on and look at how the project’s goal will solve that problem. We use a three-pronged approach to writing the goal statement, with the three prongs being purpose, advantage, and measurement (PAM).

## Purpose

The problem is that ice on the roads causes accidents. The only viable<sup>2</sup> solution to this problem is to treat the roads to prevent the ice from forming (and presumably to melt the ice if it has already formed). Thus you can write the purpose for this project as follows:

*Purpose: To accurately forecast road freezing and schedule de-icing treatment.*

The purpose of the project should be not only to solve the problem, but also to provide a business advantage. Naturally, if there is an advantage, you must be able to measure it.

## FOOTNOTE 2

There are other solutions, but none of them are viable. Roads could be heated (expensive), roads could be closed (unpopular), motorists could be asked to fit snow chains (unlikely they would comply), or drivers could be asked to learn ice-driving skills (unbelievable).

---

*The purpose of the project is not only to solve the problem, but also to provide a business advantage to the owner of the product created through the project.*

---

## Advantage

The business advantage is the reduction—ideally the elimination—of accidents due to ice. The road authorities (your owners) have a charter to maintain their roads in a condition that will make for safe motoring. Thus the owner gets the following advantage from the product:

*Advantage: To reduce road accidents by eliminating icy road conditions.*

## Measurement

Is this advantage measurable? It can be. The success of your product can be measured by the reduction in the number of accidents where ice is a contributing factor:

*Measurement: Accidents attributed to ice shall be no more than one accident per 10,000 vehicle-miles traveled on all roads in the districts covered by the product.*

You have stated a measurable goal, and monitoring the accidents for a winter or two is reasonable. Accident statistics, police reports, and road usage data are already being collected, so you should have no trouble finding data to measure the performance of your product and establish whether it is successful.

But is this a reasonable goal? Is the elimination of most of the accidents due to ice worth the cost and effort of building the product? And where did “one accident per 10,000 vehicle-miles” come from? The Northumberland County Highways Department representative (your initial customer) at the blastoff tells you that this is a target figure set by central government. If it can be achieved the county government will be satisfied, and county officials are prepared to spend money to achieve this target.

Is it viable? One of the reasons for having a blastoff meeting with the key stakeholders present is to answer questions like this. One of the stakeholders is from the National Road Users Association; she assures you that this group’s research shows ice treatment is effective and the expected outcome is realistic.

Is it achievable? The stakeholders representing the product designers and builders, the technical experts from the hardware side, and the meteorologist all assure the blastoff participants that the technology is available, or can be built, and that similar software solutions are known by the team.



Note the major aspects of the project goal:

Purpose: What should the product do?

Advantage: Which business advantage does it provide?

Measurement: How do you measure the advantage?

Viable: Given what you understand about the constraints, is it possible for the product to achieve the business advantage?

Feasible: Given what you have learned from the blastoff, is it possible to build a product to achieve the measure?

Achievable: Does the organization have (or can it acquire) the skills to build the product and operate it once built?

Sometimes projects have more than one purpose statement. Look at the customer's statement:

"Road treatment is sometimes indiscriminate, and this wastes de-icing material and causes environmental damage."

This reveals another purpose for the project:

*Purpose: To save money on winter road de-icing costs.*

The advantage stemming from this purpose is that accurate forecasting reduces the cost of treatment—only roads in imminent danger of freezing are treated. Additionally, by preventing ice from forming on road surfaces, damage to roads is reduced.<sup>3</sup>

The advantage is straightforward:

*Advantage: Reduced de-icing and road maintenance costs.*

The measurement of reduced costs is always the amount of money that can be saved:

*Measurement: The cost of de-icing shall be reduced by 25 percent of the current cost of road treatment, and damage to roads from ice shall be reduced by 50 percent.*

Naturally, you need to know the current costs and damage expenditures so that you will know when they have been reduced by 25 percent and 50 percent, respectively. If there is supporting material available, then cite it in your specification:

#### FOOTNOTE 3

When the water lying in cracks in the road surface freezes, it expands and forces the crack to expand. Eventually, this process results in significant fractures and holes in the road surface.

*Supporting Materials: Thornes, J. E. "Cost-Effective Snow and Ice Control for the Nineties." Third International Symposium on Snow and Ice Control Technology, Minneapolis, Minnesota, Vol. 1, Paper 24, 1992.*

The engineers also know that applying too much salt compounds to roads damages the environment. By having a more accurate treatment, less material finds its way to the environs of the roads, and less damage results. This means that accurate forecasting gives you another advantage:

*Advantage: To reduce damage to the environment by unnecessary application of de-icing compounds.*

This advantage can be measured by comparing the amount of de-icing material used by the product with that used at present:

*Measurement: The amount of de-icing chemicals needed to de-ice the authority's roads shall be reduced to 70 percent of current usage.*

*Supporting Materials: Thornes, J. E. "Salt of the Earth." Surveyor Magazine, December 8, 1994, pp. 16–18.*

Note that the purpose statement results in an advantage and a measurement. If you cannot express an advantage for the purpose, or if the advantage is not measurable, then it should not be part of your specification. For example, suppose the purpose of a project is something vague:

*Purpose: To improve the way we do business.*

The advantage here is unclear. Does your owner want the business to make more money, or does he want the business processes to function more smoothly? Or something else? The discipline necessary to give the purpose an advantage and a measurement means that fuzzy or ill-defined purposes are far less likely to find their way into your specifications.

You cannot build the right product unless you know precisely what the product is intended to do and how the product's success is to be measured. Whether the organization using the product achieves the target defined by the product purpose may depend on how it uses the product. Obviously, if the product is not used as intended, then it may fail to provide the advantages for which it was built. Thus the statement of project purpose must assume that the resulting product will be used as intended.

---

*You cannot build the right product unless you know precisely what the product is intended to do and how the product's success is to be measured.*

---

When you write the goal, you should make the point that all decisions about the project are driven by it. Make sure everyone understands that if the goal changes during the project, then so do the scope, stakeholders, and any requirements that have already been defined.

## Constraints

Constraints (which appear in the early part of the requirements specification template) are global requirements. These restrictions help determine which subset of requirements can be included in the eventual product. Constraints affect decisions about the scope of the product by limiting the amount of time or money that may be spent on the project. Sometimes the constraint is a pre-ordained design decision that limits the way the problem is solved.


You can think of constraints as a special type of requirement that provides some guidance on where to focus your requirements-gathering efforts. These limitations are written as if they were regular requirements. Your management, your marketing colleagues, or your sponsor probably already knows the constraints—the task at blastoff time is to elicit and record them.

### Solution Constraints

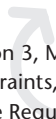
Your specification should describe any mandated designs or solutions. For example, your management may tell you that the only acceptable solution is one that will run on a tablet device—no other design is allowable. While we caution you against designing the solution before knowing the requirements, perhaps for some overriding reason—marketing, cultural, managerial, political, customer expectations, or financial—only one acceptable design solution exists. If this is the case, then that item should be included in your requirements as a constraint.

Any partner or collaborative applications should also be brought to light and recorded at this time. These are other applications or systems with which your product must cooperate. For example, your product may have to interact with existing databases, reporting systems, or Web-based systems; thus the interfaces to those systems become constraints on your product. Mandated operating systems should also be included in this section of the template.

Off-the-shelf and open-source applications, if they are to be used, or interacted with, are recorded under the “Constraints” heading as well. There may be good reasons for mandating this cooperation—but then again, there may not. Along with your stakeholders, consider whether the decision to incorporate ready-built software is appropriate for your situation.



See Chapter 13, The Quality Gateway, for more on using the project purpose as a test for relevancy. The Quality Gateway runs each requirement through a series of tests, including relevancy. If a requirement is not in some way relevant to the purpose, it is rejected.



Section 3, Mandated Constraints, of the Volere Requirements Specification Template provides a description of how you record constraints. You can find the template in Appendix A.

## Project Constraints

Project constraints describe the time and financial budgets for the project. These parameters should be known at scoping time, because they affect the requirements you gather later. If you have a \$500,000 budget, there is no point in collecting the requirements for a \$1 million product.

Time constraints can be imposed to enable the product to meet a window of opportunity, to coincide with coordinated releases of associated products, to meet the scheduled start-up of a new business venture, or to satisfy many other scheduling demands. If this type of constraint exists, then you should make your team aware of it. Keep in mind that a time constraint is not the same thing as an estimate of the time necessary to complete the project.

Financial constraints indicate how extensive the product may be. They also give you a good indication as to whether the product is really desired by the putative customers. If the budget is impossibly small, it probably indicates that the priority for the project is so low that no one really wants the product. Impossibly small budgets and impossibly short deadlines almost always cripple projects—and there is no reason to think your project would be different.

---

*Financial constraints indicate how elaborate the product may be, and they give you a good idea if the product is really wanted.*

---

## Naming Conventions and Definitions

Names are important. Good names convey meaning; poor names do the opposite. Additionally, we have found that every project has names that are particular to it, and this terminology should be recorded to make communication easier, and future understanding more reliable. During the scoping time you begin to collect and record the names, along with their agreed-upon meanings.

Record the names in Section 4, Naming Conventions and Terminology, of the specification template. This glossary serves as a reference point for the entire project. We are always amazed at how many misunderstandings occur simply because no central glossary is available, and how effective good names can be at communicating meaning. It is worth expending effort in this area to ensure smooth communication later on in the project.

For example, the IceBreaker project team added the following definition to their glossary during the blastoff:

*Weather station Hardware capable of collecting and transmitting road temperature, air temperature, humidity, and precipitation readings. Weather stations are installed in eight locations in Northumberland.<sup>4</sup>*

### FOOTNOTE 4

Weather stations are also used as ice detection systems on airport runways.

---

*Every project has names that are peculiar to it.*

---

Starting to define terminology at scoping time has a distinct advantage: You make the words visible. The stakeholders can then discuss them and change them to reflect their consensus of the meaning. Subsequent development activities build on understanding the terminology more precisely and use it as the basis for building a complete data dictionary—see Section 7 of the template.

---

*Starting to define terminology at scoping time has a distinct advantage: You make the words visible. The stakeholders are able to discuss and change them to reflect the consensus of the meaning.*

---

## How Much Is This Going to Cost?

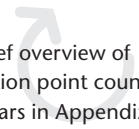
At this point, you have a fair amount of information on which to base your estimates of cost and effort. The needed effort is usually proportional to the amount of functionality contained within the work area, and this relationship makes sense—the more functions done by the work area, the more effort needed to study it and devise a solution.

At this stage you do not know the size of the product—how much functionality it will contain. Also, given the huge variations in the cost of implementing various technological solutions, you are well advised not to attempt (yet) to estimate the complete development cost. However, you know the size (in terms of its functionality) of the work area—or you do if you measure it.


The easiest way to measure the size or functionality of the work area is to count the number of adjacent systems on the context model as well as the number of inputs and outputs. While more accurate ways to measure size have been devised, counting the inputs and outputs is a quick technique that gives you a far better idea of size than merely guessing. If your context has more than 30 inputs and outputs, then it falls into the “average cost per input/output” range of estimating. Your organization has an average cost for gathering the requirements for one input or output. You can determine this cost by going back to previous projects, counting the number of inputs and outputs on the context diagram, and dividing this number into the total cost of that requirements investigation.

A more accurate estimate can be developed by determining the number of business events that affect the work. The number of business events can be identified by inspecting the context diagram. Each business event has an amount of functionality that responds to it, so the number of business events is the determining factor in the cost of the requirements effort. It is, of course, necessary to know the cost to your organization of analyzing the average business event: You can learn this cost by looking at previous projects or, if necessary, running a benchmark. Multiply the cost per event by the number of events to give a reasonably accurate cost for requirements gathering.

More accurate still is function point counting. At this stage you need to have an idea of the data stored by the work. This can usually be identified in a short time if your team includes some experienced data modelers. Function point counting measures the amount and the complexity of the data processed by the work—the inputs and the outputs from the context



A brief overview of function point counting appears in Appendix C, Function Point Counting: A Simplified Introduction.

**READING**


Bundschuh, Manfred, and Carol Dekkers. *The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement*. Springer, 2010.

---

*The key consideration is not so much that you use a particular estimating system, but rather that you use a system based on measurement, not hysterical optimism.*

---



---

*Have you ever worked on a project where nothing went wrong?*

---

model—along with data stored within the work. Enough is known about function points to enable you to find figures for the average cost per function point of requirements investigation for your particular industry.

It really doesn't matter which estimating system you use, but it does matter that you use a system based on measurement, not one based on hysterical optimism. Too much risk is courted by not measuring; there is too much evidence to support the downside of not measuring, and too much known about measuring, to have any excuse for not doing it.

We strongly advocate that all projects take into account the amount of functionality in the work area that they are about to improve. Although some development techniques shun this upfront measurement in favor of time boxing, there is a strong argument to be made for measuring the amount of functionality to be delivered by each iteration. It is always helpful to project management, and the requirements activity, to know (rather than guess) how much effort lies ahead.

## Risks

We face risks every day. Just leaving your home to go to work involves some risk—your car won't start, the train will be late, you will be assigned to share an office with a boring person with body odor problems. But still you go to work each day, because you know the risks and consider the outcome (a pay packet or job satisfaction) to be worth the risk. Of course, once you are at work, you might well plunge into projects where you have no idea of the risks involved, and thus no idea whether the outcome is worth braving the risks.

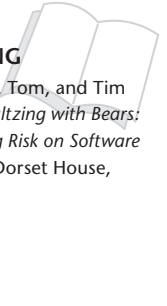
Have you ever worked on a project where nothing went wrong? No? Nor has anyone else; something always goes wrong. But did you ever try and figure out ahead of time what could go wrong, and do something to prevent it from going wrong, or at least allow for the mishaps by budgeting for them? This, in its simplest sense, is risk management.

Your blastoff process should include a short risk assessment. Such an assessment is probably outside the remit of the business analyst, and should be done by a competent risk-assessment person. The job is to assess both those risks that are most likely to happen and those risks that will have the greatest impact if they do, in fact, become problems. The deliverables from your blastoff provide input for the risk assessor to identify risks. For each identified risk, the assessor determines the probability of it becoming a problem, along with its cost or schedule impact. At the same time, the assessor determines the early-warning signs—the happenings or conditions that signal a risk is coming to fruition. In some cases where the risks are considered serious, a risk manager is assigned to monitor for the telltale signs that some risks are about to become full-blown problems.

Risk management is common-sense project management or, in the words of our partner Tim Lister, “project management for adults.” If your organization is not doing it, then you should prepare for the budget or time overruns that are coming your way. The most noticeable effect of doing risk analysis is that it makes the risks visible to all stakeholders. Once aware of the risks, they can contribute to risk mitigation. Similarly, the risk assessor makes management aware of the risks and their impact if they become problems.

#### READING

DeMarco, Tom, and Tim Lister. *Waltzing with Bears: Managing Risk on Software Projects*. Dorset House, 2003.



## To Go or Not to Go

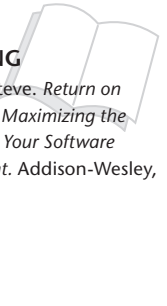
The deliverables from the project blastoff indicate the viability of your project. When you take a hard look at what these deliverables are telling you, you can decide whether it makes good business sense to press the button and launch the requirements project.

Consider your deliverables:

- Is the product goal clear and unambiguous? Or does it contain fudge words?
- Is the goal measurable? That is, will it give a clear indication when you have successfully completed the project?
- Does the goal indicate an actual benefit to the owner?
- Is it viable? Is it possible to achieve the objectives of the project within the allotted time and budget?
- Have you reached agreement on the scope of the work?
- Are there some risks that have a high probability of becoming problems?
- Is the impact of these risks such that it makes the project unfeasible?
- Is the cost of investigation reasonable given the product's benefit?
- Are the stakeholders willing to be involved?
- Do you have sufficient justification to invest in the project?
- Do you have enough reasons not to invest in the project?
- Is there any further investigation that you should do before launching the requirements project?

#### READING

Tockey, Steve. *Return on Software: Maximizing the Return on Your Software Investment*. Addison-Wesley, 2004.



The point is to make an objective decision based on facts, not on boundless enthusiasm or giddy optimism. In the book co-authored with our Guild partners, *Adrenaline Junkies and Template Zombies*, one of the essays is called “Dead Fish.” A Dead Fish project is one of those where it is known from the moment of inception that it is doomed to failure, and yet no one on the project stands up to say that it is doomed, that it will hang around, smelling

“From Day One, the project has no chance of meeting its goals; most people on the project know this and say nothing.”

—Adrenaline Junkies and  
Template Zombies: Understanding  
Patterns of Project Behavior  
(Dorset House, 2008)

badly, long after it should have been thrown out with the other trash. Sadly, Dead Fish projects are all too common—and we urge you to do whatever it takes to not be part of one. A little consideration at this stage can prevent Dead Fish projects from being started; it can also give good projects a flying start.

Other project management techniques can also be brought into play at this time. Sadly, they all have cheesy acronyms, which makes it a little hard for your authors to take them too seriously:

### SWOT

The Strengths, Weaknesses, Opportunities, and Threats are listed. These factors are used to evaluate the overall value and risk of the project.

### ALUo

The Advantages, Limitations, Unique Qualities, and overcome limitations of proposed options. This technique comes from The Creative Problem Solving Group.

### SMART

The project must be Specific, Measurable, Attainable, Relevant, and Time-bound. Management considers if the project is all of these things.

### PESTLE

This model looks at the Political, Economic, Sociological, Technological, Legal, and Environmental factors of the project. It is often used in conjunction with SWOT.

### CATWOE

This technique comes from Peter Checkland's soft systems methodology and means that you consider the Customers, Actors, Transformation Processes, World view, Owners, and Environment for the project.

Each of these techniques has its adherents, and when used correctly all of them may provide some value. Our intention here is not to discuss these approaches, but rather to provide a pointer to techniques that can be used in conjunction with the deliverables from the blastoff meeting.

## Blastoff Meetings

We suggest that the key stakeholders get together for a day or so and derive the deliverables discussed in this chapter. We understand that in many



organizations this type of meeting, despite its merit, is simply not possible. Nevertheless, there are other ways to achieve the same results.

While coming together is important, it is the deliverables that really matter. Some organizations come up with these items in other forms—a lot of companies write a business plan or some similarly named document that covers many of the topics we advocate. This is fine as long as you have an objective, quantifiable plan and it is circulated and agreed to by the stakeholders.

Some organizations use feasibility studies as a way of getting their projects started. Of course, the feasibility study must take an honest look at the costs and risks as well as the benefits from the product. Provided the study delivers realistic numbers, it will serve. We make the proviso that all the key stakeholders must have seen and commented on the accuracy of the feasibility study. You don't have to hold a meeting, but you do need to know all the facts that the meeting would deliver.

---

*You don't have to hold a meeting, but you do need to know all the facts that the meeting would deliver.*

---

## Summary

The Project Blastoff is about knowing: Knowing what you want the product to do for you, and what it will cost to build it. Knowing the scope of the work that is to be studied so as to gather the requirements for the product. Knowing which people will be involved in the project, and having them know what is expected of them. Knowing the users, which in turn will lead you to knowing the usability requirements for the product. Knowing the constraints on the project—how much money have you got to spend, and how much or how little time do you have to deliver the product? Knowing the words to be used on the project. Knowing whether you can succeed.

The blastoff delivers knowledge at a time that it is most useful. It is at the beginning of the project that crucial decisions—decisions that affect all subsequent stages of the project—must be made. If they are made badly, the project will suffer; if they are made well—and there is no real reason why all decisions cannot be good ones—the project will prosper.

The blastoff deliverables will reappear from time to time in this book. Some of them are used as input to the mainstream requirements activities; none of them is wasted.

---

*The Project Blastoff is about knowing.*

---

*This page intentionally left blank*

# Business Use Cases

*in which we discuss a fail-safe way of  
partitioning the work and so smooth the way  
for your requirements investigation*



The blastoff process, which we described in the previous chapter, establishes the scope of the work—the business area to be studied. This scope—ideally shown graphically as a context diagram—defines a business area, part of which is to be automated by your intended product. In reality, this work scope is probably too large to be studied as a single unit. Just as you cut your food into small bites before attempting to eat it, so it is necessary to partition the work into manageable pieces before studying it to find the product’s requirements.

In this chapter we provide heuristics for finding the most appropriate use cases. In later chapters you will see how this process allows you to arrive at the most relevant and useful product to build.

“Never eat  
anything bigger than  
your head.”

—B. Kliban

## Understanding the Work

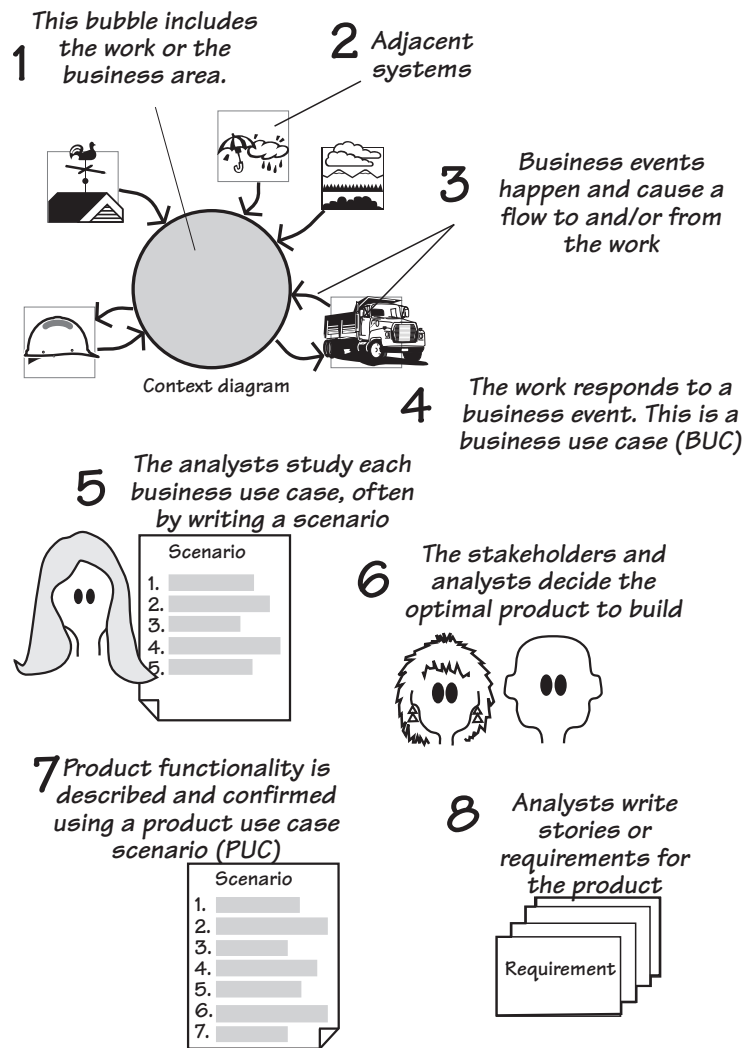
The product you intend to build must **improve its owner’s work**; it will be installed in the owner’s area of business and will do part of (sometimes all of) the work. It does not matter which kind of work it is—commercial, scientific, embedded real-time, manual, or automated—you always have to understand it before you can decide which kind of product will best help with it.

When we say “work,” we mean the system for doing business. This system includes the human tasks, the software systems, the machines, and the low-tech devices such as telephones, photocopiers, manual files, and notebooks—in fact, **anything that is used to produce the owner’s goods, services, or information**. Until you understand this work and its desired outcomes, you cannot know which product will be optimally valuable to the owner.

Figure 4.1 presents an overview of how we intend to proceed. You might wish to refer back to this figure while reading the text—it will help smooth the way.

Figure 4.1

The scope of the business problem—the work—is agreed to by the blastoff participants. The scope defines both the work area to be studied and the adjacent systems that surround it. The adjacent systems supply data to the work and/or receive data from it. Business events happen in the adjacent systems—usually the event produces a demand for a service provided by the work. In addition, time-triggered business events occur when it is time for the work to provide some information to the adjacent system. The response the work makes to the business event is called a business use case; it includes all of the processes and data necessary to make the correct response. The requirements analysts study the functionality and data of the business use case with the help of the appropriate stakeholders. From this study, they determine the optimal product to build, and construct a product use case scenario to show how the actor and the product will interact. Once agreement is reached on this product use case scenario, the requirements analysts write the requirements or the user stories for the product.



If we have some systematic and observable way of partitioning the work, then we are far more likely to be consistent with the results we get. We turn to *business events* as our preferred way of partitioning. The responses to the *business events*—we call these *business use cases* (BUCs)—meet the following criteria:

- They are “natural” partitions—each one makes an obvious and logical contribution to the work.
- They have minimal connectivity to other parts of the work.
- They have a clearly defined scope.
- They have rules for defining their scope.

- They have boundaries that can be observed and defined.
- They can be named using names that are recognizable to stakeholders.
- Their existence can be readily determined.
- They have one or more stakeholders who are experts for that part of the work.

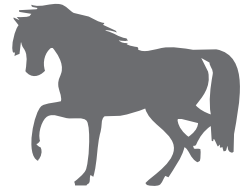
Before we examine business use cases, let's look at how different projects need them.

## Formality Guide

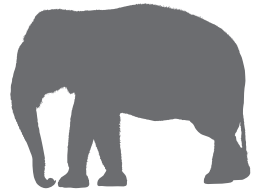
Rabbit projects should pay particular attention to this chapter. When running an iterative project, it is important to have a rock-solid grasp of the business problem to be solved. We strongly suggest that rabbits make use of business use cases to explore their problem domain before starting to formulate a solution. This approach does not add to the documentation load, and it lessens the time spent delivering inappropriate solutions.



Horse projects should consider partitioning the work area using business use cases as we describe them in this chapter. We have found that the BUC scenarios are a useful working tool for discussing the current and future work with your stakeholders. There is also the possibility of using BUC (and later product use case [PUC]) scenarios as the documentation to pass along to the developers, which allows you to avoid writing many of the detailed requirements. We shall discuss this aspect in later chapters.




Elephant projects should definitely use business events. Given that elephant projects have a large number of stakeholders, clear communication is both important and difficult. We have found that BUC scenarios are an ideal mechanism for discussing the work in geographically distributed teams. Later, the BUC scenarios and their PUC derivatives are maintained as part of the formal documentation. The BUC scenario is also useful for discussing high-level issues with outsourcers.



## Use Cases and Their Scope

The term *use case* was coined by Ivar Jacobson back in 1987 as a way to describe an interaction between a system and a user of that system. Jacobson needed to break the system into smaller units, as he felt that object models were not scalable. Thus, to conquer the complexity and largeness of modern systems, he said it was first necessary to partition them into convenient chunks, and that these chunks should be based on the user's view of the system.

**READING**


Jacobson, Ivar, et al.  
*Object-Oriented Software  
 Engineering: A Use Case  
 Driven Approach*. Addison-  
 Wesley, 1992.

---

*If the responsibilities of the actor and the system are established at the beginning of the analysis process, and the requirements gathering focuses on the automated system, how will you ever understand the work the actor is doing or the work the automated product could be doing for the actor?*

---

However, Jacobson left us with some loose ends. For example, his definition of a use case does not indicate precisely where or how a use case starts and ends. In fact, Jacobson's definition of a use case must have left some ambiguity; other authors have written about use cases and very few of them have the same idea of what it is. There are about 40 published definitions of "use case" with almost none of them agreeing. This chaos is unfortunate.

Jacobson also uses the term *actor* to mean a user role, or perhaps another system, that lies outside the scope of the system. The *system* in this usage is presumed to be the automated system under construction (we refer to this item as the *product*). This now leads to a question: How can anyone know what the automated system is to be before having an understanding of the work for which this system is to be used?

Think about this: If the responsibilities of the actor and the system are established at the *beginning* of the analysis process, and the requirements gathering focuses on the automated system, how could you ever understand the work the actor is doing, or the work the automated product might be doing for the actor? Failure to understand the actor's true task—which surely happens if you exclude the actor from the analysis study—means that you run the risk of missing opportunities for automation, or automating where a non-automation would be a better solution. You also could be guilty of building products that are not as useful as they might be as well as run the risk of constructing interfaces that ultimately do not satisfy what the actor is really trying to do.

We can rush headlong into a solution, or we can find out what the problem is. The problem is simply the work that you are meant to improve, and the first step is to establish the scope or the extent of this work. If it is to be effective, this work scope must *include* the intended actors and all the work they are doing. Once you have established a satisfactory scope for the work, you partition it into smaller pieces; these pieces are the business use cases.

This chapter discusses the process of partitioning the work. Keep in mind as you are reading it that it probably takes longer to describe each of the steps than it does to actually complete most of them.

## The Scope of the Work

The work is the business activity of the owner of the eventual product; alternatively, you can think of it as the part of the business that your customer or client wants to improve. To understand this work, it is best to think about how it relates to the world outside it. This perspective makes sense because the work exists to provide services to the outside world. To do so, the work must receive information and signals from the outside world, use these

---

*Your context diagram shows how your work relates to its business environment.*

---

inputs as raw materials, and send information and signals back to the outside world—the customer for such things. This outside world is represented by *adjacent systems*, which comprise the automated systems, people, departments, organizations, and other parties that place some kind of demand on, or make some kind of contribution to, the work.

To locate your work within the outside world, you demonstrate its context by showing how the work connects to the adjacent systems. In other words, your context diagram shows your work in its business environment.

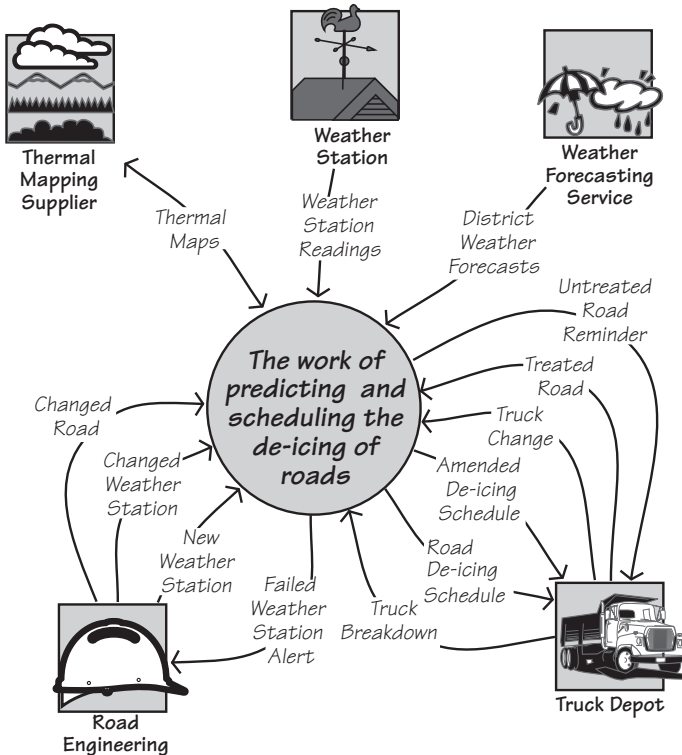
Figure 4.2 shows the context diagram for the work of predicting when roads are due to freeze and scheduling trucks to treat them with de-icing material (this diagram also appeared in Chapter 3). The work is surrounded by adjacent systems that either supply the data necessary for this work, or receive services and information from the work.

The work to be studied must include anything that can be affected by your product. If you are building a product intended to automate part of some existing work, then the scope of the study should include those parts of the existing work—human activities, together with any existing computer systems—that could potentially be changed by the eventual product. For embedded systems, there may be no human activity in your work, but the work scope must include any devices that can be changed or somehow

---

*The work to be studied must include anything that can be affected by your product.*

---



**Figure 4.2**

The context diagram showing the scope of the work. The central area of the diagram represents the work you are about to study, and the product you eventually build becomes part of this work. The outside world is represented by the adjacent systems—Weather Station, Truck Depot, and so on. The named arrows represent flows of information between the adjacent systems and the work.

affected by the current development. Even if you are building an electro-mechanical device, such as an automated teller machine (ATM), and most of the human participation occurs outside the product boundary, your work context must still include the work that the human will be doing with the device.

---

*The work scope includes anything that you are permitted to change, plus anything that you need to understand to decide what can or should be changed.*

---

While we are talking about the work context diagram, note the limited, but nevertheless crucial, aim of the model. This model shows only the flows of information. It does not attempt to show the constraints upon the work, although these limitations may be inferred from the model. Likewise, it does not explicitly show who or what is doing the work, although this information might also be inferred. As is true of most models, the context model is an abstraction that shows a single view. In this case, by highlighting the flows of information, we are able to make better use of the model for determining the business events affecting the work. But first, let's look at one part of the context model in more detail: the outside world.

### The Outside World

As we saw earlier, the adjacent systems are those parts of the world that connect to the work by delivering data to it or receiving data from it.

Adjacent systems behave like any other systems: They contain processes and consume and/or produce data. You are interested in them because they are often customers for the information or services provided by your work, or because they supply information needed by your work. You can see these relationships by looking at the data flows on the context diagram. It is through these informational connections that the adjacent systems influence the work.

---

*Within reason, the farther away from the anticipated automated system you look, the more useful and innovative your product is likely to be.*

---

To find the adjacent systems, you sometimes have to venture outside your own organization. Go to the customers for your organization's products or services. Go to the outside automated systems and organizations that supply information or services to your work. Go to the other departments that have connections to the work. Use the guideline that the farther away from the anticipated automated system you look, the more useful and innovative your product is likely to be.

---

*Do not be limited by what you think might be the limits of a computer system. Instead, try to find the farthest practical extent of any influence on the work.*

---

You will usually find that your work is also closely connected to one or more computer systems, often within your own organization, or that you are making an enhancement to an existing computer system. In this case, the computer systems, or the parts that you are not changing, are adjacent systems. The interfaces between your work and the existing computer systems are critical. Although they may prove difficult to describe, you can never know the extent of your work, and eventually the extent of your product, if you do not define these interfaces clearly.

Think of it this way: The adjacent systems are the reason why the work exists; they are customers for the services produced by the work. The work



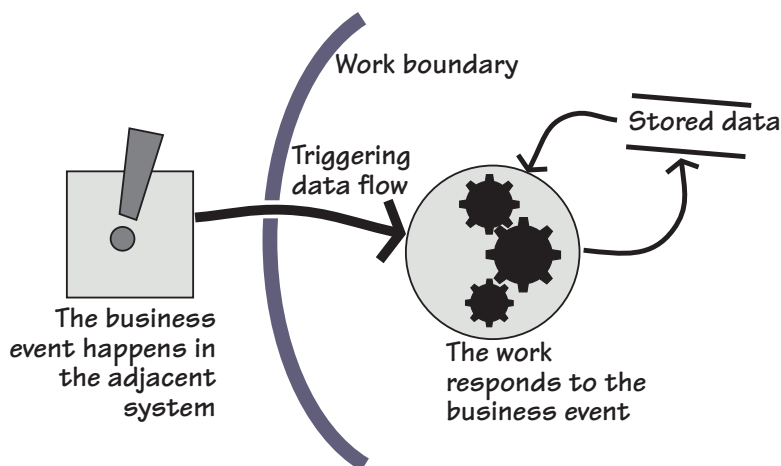
produces these services either on demand or at prearranged times, and when it does so, the work is responding to a business event.

## Business Events

Any piece of work responds to things that happen outside it—it's that simple. For the sake of clarity, because we are discussing your owner's work or business, we shall call these happenings *business events*.

Let's look at an example of a business event. You are reading this book, so your authors sincerely hope you paid for it. Let us for the moment suppose that you bought it online. You had already found the book, looked at the sample pages, and decided that you wanted it. At that moment, one of two things could have happened: (1) You could have decided that you had something other to do and abandoned the transaction or (2) you could have decided to buy the book. That moment—that instant when you decided to buy the book—is the business event. Of course, just deciding that you want the book is not quite enough; you have to tell the work that you want it.

You signaled the work by indicating that you wanted to check out (or whatever mechanism your online bookseller has for such things) and provided a credit card and shipping address and so on. This incoming flow of data triggered a response in the bookseller's work, which was to transfer ownership of the book to you, debit your credit card, and send a message to the fulfillment department to ship the book to you. If you bought the book as an e-book, that last step would be replaced by one that started the download.



**Figure 4.3**

Business events and their responses: A business event happens at the moment the adjacent system decides to do something, or as part of its work some processing condition occurs. The adjacent system tells the work that the event has happened by sending a triggering data flow. When this stream of data arrives, the work responds by processing the incoming data, and by retrieving and recording stored data.

Another example: You pay your credit card bill at the end of the month—that is a business event as seen from the point of view of the credit card company. The credit card company responds to this event by checking that your address has not changed and then recording the date and amount of your payment.

In these examples, the moment you decide to buy the book and the moment you decide to pay your monthly bill are the business events. There is always some data resulting from the business event (the triggering data flow), which invokes a preplanned response to the event. This response is the business use case.

You as a customer do not own or control either the bookshop or the credit card company. Nevertheless, in both examples you did something to make them respond—they did some processing and manipulated some data. Think of it this way: Those pieces of work have a preplanned business use case that is activated whenever an outside body (the adjacent system) initiates a business event. Figure 4.4 illustrates this idea.

---

*When a business event happens, the work responds by initiating a business use case.*

---

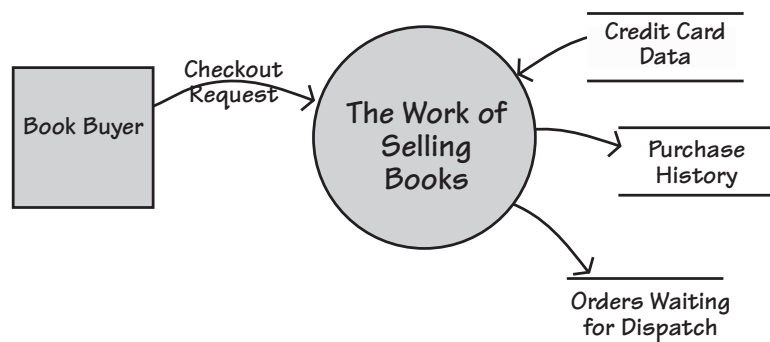
Note that business use cases are triggered by the arrival of a data flow from the adjacent system. In the preceding examples, these data flows were your request to buy the book and your payment slip and check arriving at the credit card company. As a consequence, the responsibility for triggering the business use case lies outside the control of the work. We will return to this point shortly. First, however, let's look at another kind of business event.

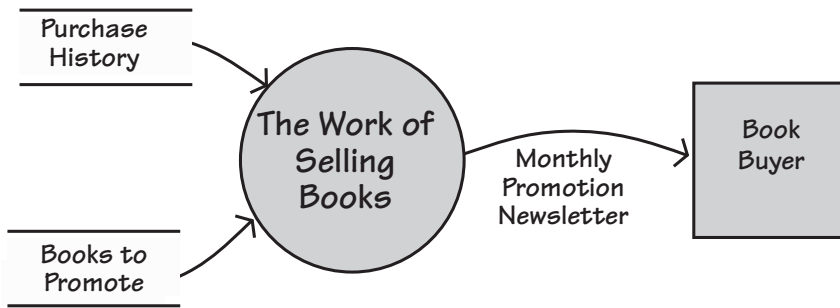
### Time-Triggered Business Events

Time-triggered business events are initiated by the arrival of a predetermined time (or date). For example, your insurance company sends you a renewal notice a month before the anniversary of your policy; your bank sends you a statement on an agreed day of every month. “The arrival of a predetermined time” may also mean that a certain amount of time has elapsed since another event happened. For example, a computer operating

**Figure 4.4**

A business event takes place outside the scope of the work, and the work learns that it has happened through the arrival of an incoming flow of information. The work contains a business use case that responds to this business event.



**Figure 4.5**

A time-triggered business event happens when a prearranged time is reached. This is based on either a periodic occurrence (for example, the end of the month, or a certain time each day), a fixed time interval (for example, three hours since the last occurrence), or a certain amount of time elapsing since another business event (for example, 30 days after sending out an invoice). The normal response is to retrieve stored data, process it, and send some information to an adjacent system.

system may check the available memory 2.4 microseconds after the last time it checked, or you may be sent a reminder that you borrowed a library book six weeks ago.

The usual response to a time-triggered business event is to retrieve previously stored data, process it, and send the resultant information to an adjacent system. Consider the example depicted in Figure 4.5.

Once the predetermined time for the event arrives, the work's response is to do whatever is necessary to produce the output. This almost always involves the retrieval and manipulation of stored data. Once again, we use the response to the time-triggered business event—the business use case—as our unit of study.

## Why Business Events and Business Use Cases Are a Good Idea

We may seem to be going to a lot of trouble to describe something that may, at first glance, seem fairly obvious. But this care with the subject is warranted: Our experience has amply demonstrated the value of having an objective way of partitioning the work, and the value of understanding the work itself, before plunging into the solution. The result is that you discover the real requirements, and you discover them more quickly.

Your partitioning of the work will be more objective if you identify the responses to outside stimuli; after all, that is the way your customers see your business. The business's internal partitions—department and processors, whatever they may be—hold no interest for outsiders. Similarly, it is likely that the current partitioning of any system is based on technological and political decisions made at the time the system was built. Those decisions may no longer be valid—at the very least, you should question them and avoid perpetuating them just because they are there at the moment. By looking not at the inside, but from the outside, you get a clearer idea of the most functional way of partitioning the work.

---

*By looking not at the inside, but from the outside, you get a much clearer idea of the most functional way of partitioning the work.*

---

---

*The work's response  
to a business event  
brings together  
all the things that  
belong together.*

---

The work's response to a business event brings together all the things that belong together. As a result, you get cohesive partitions with minimal interfaces between the pieces. This partitioning gives you more logical chunks of work for your detailed requirements investigation—the fewer dependencies that exist between the pieces, the more the analysts can investigate the details about one piece without needing to know everything about all the other pieces.

There is one more reason for using business use cases, and that is to prompt an investigation of what is happening at the time of the business event.

### **The “System” Cannot Be Assumed**

In many of the texts available today, the existence of a “system” is assumed. That is, the author guesses what he thinks the boundary of the automated system should be, and begins the use case investigation by looking at the actor and the interaction with the automated system, and completely ignores the work surrounding this interaction. *To do so is dangerous and wrong.*

This product-centric way of looking at the problem means that you are ignoring the most important aspect of the automated system: the work that it is meant to improve. By starting with the automated system, projects take a massive gamble that they have, in fact, hit on the right solution, and that they have been able to do so without any study of the problem. Although we seem able to resist the blandishments of the salesman who claims that his particular insurance package will serve all our needs, we have an unhappy tendency to leap at the first automated solution that comes to mind. Looking inward at the solution discourages analysts from asking, “But why is this like it is?” This failure, in turn, leads to “technological fossils” being carried from one generation of a product to the next.

Consider this example: “An insurance clerk receives a claim from a car insurance policy holder and then enters the claim into the automated system.” This view encourages the requirements analyst to study the work of the clerk entering the details of the claim into the system. However, if you spend a few moments looking at the real business being done here, you will see that the claim is simply the insurance company's implementation—it is the *accident* that initiates this piece of business. Why is this point important? If you start your business investigation at the real origin of the problem—in other words, you look at what is happening at the time of the business event—you will build a better product. Perhaps it would be feasible to create a product that processes the claim in real time at the scene of the accident. A possible solution for that problem would be a smart phone app that knew where you were, and could take photos of the result of the accident along with the relevant license plates and driver details, and transmit them to the insurance company before the tow-trucks arrive.

Think of the *real* originator of the business event—in this case, it is the driver or owner of the vehicle (not the insurance clerk). What are the driver's aspirations here? He wants to have the vehicle repaired as quickly and effortlessly as possible; his goal is *not* to fill in claim forms and wait for them to be approved.

Another example: "A caller contacts the help desk. The help desk person initiates the use case by asking the caller for details of the problem and logging the call." The use case is the logging, and the actor is the help desk person. Again, this product-centric view misses the real business event—the event that started it all. In this case, it is the initiation of the call to the help desk.

Why is it important to find the real origin for the event? If you think of initiating the call as the business event, the correct response is to log the call, use caller ID to identify the caller's equipment, retrieve information on the equipment, and provide it for the help desk person.

Stepping back a little further and seeing the end-to-end business (see Figure 4.6), you would probably decide that the real business event is the malfunctioning of the caller's equipment. When you adopt this perspective, you also think of the equipment making the call for help itself. Perhaps a better understanding of the reasons underlying the malfunction might result in your new product giving the help desk operator historical information that will facilitate him in responding correctly to the call.

Stakeholders often don't ask for these requirements because they are thinking only of an assumed product. It is the task of the accomplished business analyst to look beyond that endpoint, which means understanding the intentions of the adjacent system at the moment it initiates the business event.

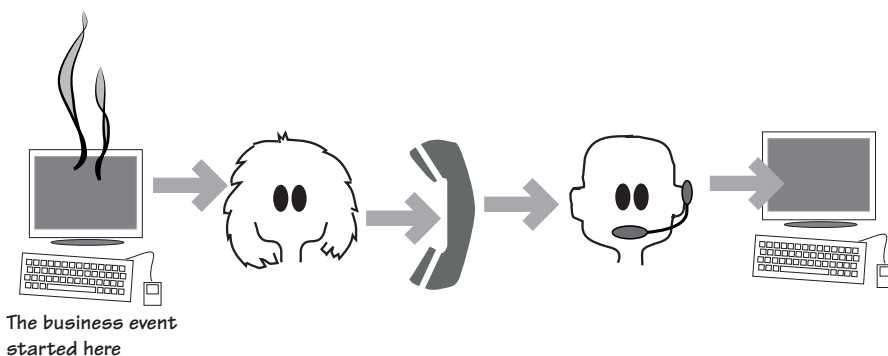
## Step Back

A security system this time: "The actor receives the shipment and logs it in." Nope. The real business event is the *dispatch* of the shipment; the business use case should log the shipment as it leaves the shipper and monitor its

---

*The requirements analyst must look past the obvious, and the current way of doing business, and instead understand the true nature of the work.*

---

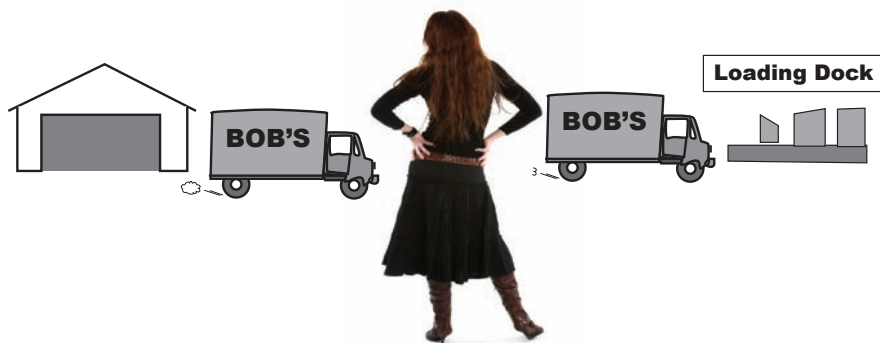


**Figure 4.6**

Consider the real end-to-end process. The aim of the diligent requirements discoverer is to look at all the activities and include them in the study. This way, the real business is revealed.

**Figure 4.7**

Step back and see the end-to-end process, and at the same time, see the real business.



transit and arrival. We need to step back and see the whole business as in Figure 4.7.

By stepping back and seeing the whole of the business process, the business analyst has the best chance of identifying the real work. Conversely, by looking only at the automated system, the analyst is often led down the path of incremental improvements that stakeholders ask for. If you can see the whole spectrum of the business use case from its real inception to its conclusion, you are in a much better position to derive a more appropriate and useful automated product.

## Finding the Business Events

Business events are things that happen and, in turn, make the work respond in some way. An event may happen outside the scope of the work (an external event), or it may happen because it is time for the work to do something (a time-triggered event). In the case of an external event, a communication to the work from the outside—represented by the adjacent systems—lets the work know that the event has happened. In the other case, the time-triggered event, the outcome is always a flow to the outside world. In either situation, whenever a business event occurs, there must be at least one data flow to show it on the context diagram.

Figure 4.8 provides the context diagram for the de-icing project. The same diagram appeared earlier in this chapter—for convenience, we repeat it here. Take a look at it and note the information flows that connect the adjacent systems to the work. For example, the flow called *Changed Road* is the result of the event that occurs when the Road Engineering department either builds a new road or makes significant alterations to an existing one. The engineering personnel advise the work of the change so that the scheduling work can update its own stored data about roads. The flow called *Road De-icing Schedule* is the outcome of a time-triggered business event: Every two hours, the work produces a schedule of the roads to be treated and sends it to the *Truck Depot*.

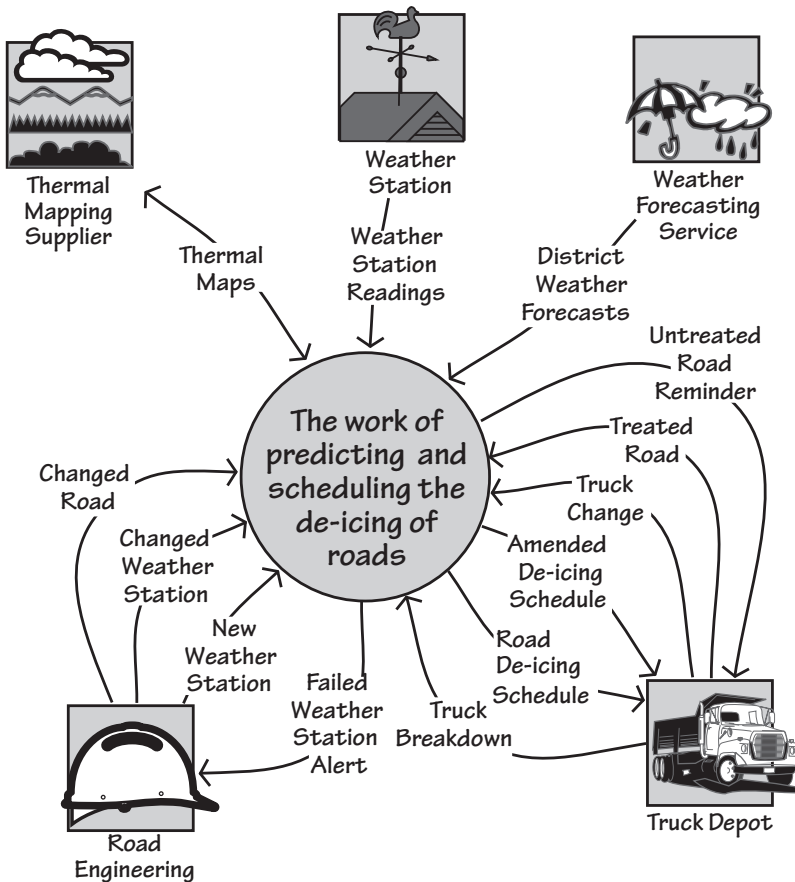


Figure 4.8

The context model for the IceBreaker work. Note the flows of data entering and leaving the work. The analyst uses these data flows to determine the business events.

Each of the flows that enters or leaves the work is the result of a business event; there can be no other reason for an external communication to exist. Looking at each flow, you can determine the business event that caused it. In some cases, several flows may be attached to the same business event. For example, when the *Truck Depot* advises that a scheduled truck has broken down or will be withdrawn from service for some other reason (the input flow is *Truck Breakdown*), the work responds. Because one of the trucks is now out of service, the other trucks have to be rescheduled to compensate for the shortfall, and the resultant outgoing flow is the *Amended De-icing Schedule*.

Table 4.1 shows a list of business events and their input and output flows for the de-icing work. Compare it with the work context diagram in Figure 4.8, and reconcile the business events with the data that flows to and/or from the work.

Admittedly, you need some knowledge of the work to figure out the business events. To this end we advise you to start the process of determining business events during blastoff, when the key stakeholders are present. In

---

*Each of the flows that enters or leaves the work is the result of a business event.*

---



**Table 4.1**  
List of Business Events  
and Their Associated  
Input and Output Flows  
for the Road De-icing  
Work

Event Name	Input and Output
1. Weather Station transmits a reading	Weather Station Readings (in)
2. Weather Bureau forecasts weather	District Weather Forecasts (in)
3. Road engineers advise there are changed roads	Changed Road (in)
4. Road Engineering installs a new weather station	New Weather Station (in)
5. Road Engineering changes the weather station	Changed Weather Station (in)
6. Time to test Weather Stations	Failed Weather Station Alert (out)
7. Truck Depot changes a truck	Truck Change (in)
8. Time to detect icy roads	Road De-icing Schedule (out)
9. Truck treats a road	Treated Road (in)
10. Truck Depot reports a problem with a truck	Truck Breakdown (in) Amended De-icing Schedule (out)
11. Time to monitor road de-icing	Untreated Road Reminder (out)

most situations, you will find the stakeholders know the business events (they may not know them by that name, but they will know what they are). If you do not identify all of the business events during the blastoff, you will see them when you begin to study the work.

## Business Use Cases

Business events are useful to partition the work, but it is the work’s *response* to the event that now captures the interest of the requirements analyst.

For every business event, there is a preplanned response to it, known as a *business use case* (BUC). The business use case is always a collection of identifiable processes, data that is retrieved and/or stored, output generated, messages sent, or some combination of these. Alternatively, we could simply say that the business use case is a unit of functionality. This unit is the basis for writing the functional and non-functional requirements (we will talk about these requirements in more detail in Chapters 10 and 11).

You can readily isolate the work of a business use case, because it has no processing connections to other BUCs; the only overlap between BUCs is their stored data. As a consequence, different analysts can investigate different parts of the work without the need for constant communication between them. This relative isolation of each business use case makes it easier to identify the stakeholders who are expert in that part of the work, and they can (with your help) describe it precisely and in detail. You can also *observe* the business use case: Business events are known to the stakeholders, and they can show you how the organization responds to any of them. For example,

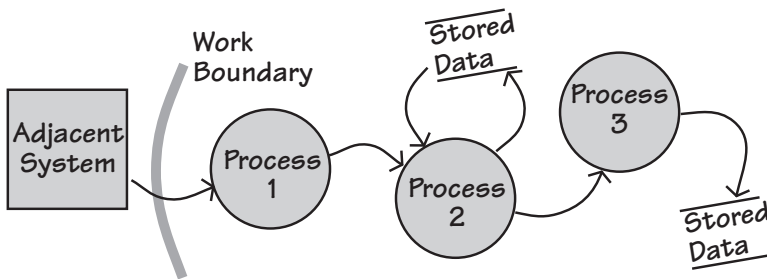
*The business use case is the most convenient unit of work to study.*

*You can identify one or more stakeholders who are expert in each event.*

it would not be hard to find someone in your favorite bookshop who can take you through the process of selling a book, or someone in your insurance company who can show you how the company processes a claim. We discuss trawling techniques for this kind of investigation in Chapters 5, 6, and 7.

The processing for a business use case is continuous—it happens in a discrete time frame. Once it is triggered, it processes everything until there is nothing left to do that can logically be done at that time. All the functionality has been carried out, all the data to be stored by the business use case has been written to the data stores, and all the adjacent systems have been notified. You can see an example of this processing in Figures 4.9 and 4.10.

The product you intend to build contributes to the work being done by the business use case. Sometimes the product will do all the work of the BUC, but usually it will do some part of it. Your product does not change the real nature of the work; it just changes the way it is done. Nevertheless, before you can design the product that makes the optimal contribution to the work, you must understand the work. Most importantly, you must understand your client's desired outcome from the work. So for the moment, forget the details and technology of the business use case, and instead look



**Figure 4.9**

The work's response to the business event is to continue processing until all active tasks (the processes) have been completed and all data retrieved or stored. You can think of the response as a chain of processes and their associated stored data. Note that the processes are surrounded by a combination of data stores and adjacent systems.



**Figure 4.10**

This model uses BPMN notation to show the same processing as in Figure 4.9, which uses data flow notation. You are urged to use whichever notation you prefer.

outside the organization to see what kind of response is needed, or wanted, by the organization's customers and suppliers.

## Business Use Cases and Product Use Cases

We have stressed the importance of understanding the work, not just the product. By looking at the larger scope of the work, you ask more questions about the business requirements and ultimately build a better product. The following example comes from a recent consulting assignment: The product is to rip a CD into MP3 or some other digital format. When the engineers looked at the technical part of the product (they are engineers, so naturally they are interested in technicalities), they saw a use case that was triggered by the insertion of the CD and then went on to rip the CD. Getting the best musical quality (the engineers were mainly concerned with achieving the desired bit rate for the transfer) seemed to be the most important thing.

We could have considered that to be the final product use case and written requirements for it. But look what happens when you step back and look at the wider context of the real business being done here (the business use case): You see something more.

What is the end user trying to accomplish? What are his aspirations and desired outcomes? We define them as “getting the music from the CD into an MP3 player.” Thus the real question is not about the technicalities of the assumed product, which is ripping CDs and converting them to MP3 format, but rather about the remainder of the true business.

Part of that business indicates that the end user wants the track names to appear on his MP3 player. Thus adding track names must be part of the BUC, as must adding images of the album cover and artist. The business use case should also allow for the order of tracks to be changed, unwanted tracks to be deleted, and any other organizational changes to the music to be carried out.

By stepping back and looking at the work being done, we were able to find a much better product to build.

So what do you have? At the outset, you have the scope of the work being studied, and the scope is bounded by the communications with the adjacent systems surrounding the work. Business events happen in the adjacent systems when they decide they want some information or service from the work, or they want to send some information to the work. Once the business event has happened and the resulting data flow has reached the work, the work responds. This response constitutes the business use case.

Study the business use case, considering what the work does and what the adjacent system desires or needs. In other words, consider whether the organization is making the correct response to the adjacent system. Once you understand the correct work of the business use case, determine the scope

---

*What is the end user trying to accomplish? What are his aspirations and desired outcomes?*

---

of the product that best contributes to that business use case. As part of this effort, consider whether the adjacent system is capable (or desirous) of making a different contribution to the work than it currently does.

Don't assume the responsibilities of the product and the adjacent systems at the beginning of the project. Instead, derive them from an understanding of the work and from what the external customer considers to be a useful product.

When that is understood, decide how much of the BUC is to be done by the product use case. Specifically, the part of the BUC handled by the automated system is the *product use case* (PUC). Sometimes the functionality of the PUC ends up being the same as that of the BUC (you have decided to automate everything), but often some part of the functionality does not become part of the PUC; you decide that this task is best done by humans.

So here's the thing: the PUC is *derived*. It is not assumed at the beginning of the requirements investigation, but rather carefully arrived at by examining the work. By deriving the PUC from the BUC, you find a more useful product, one that gives better value to its owner. And that surely is the point of your project.

The relationship between PUCs and BUCs is shown in Figure 4.11.

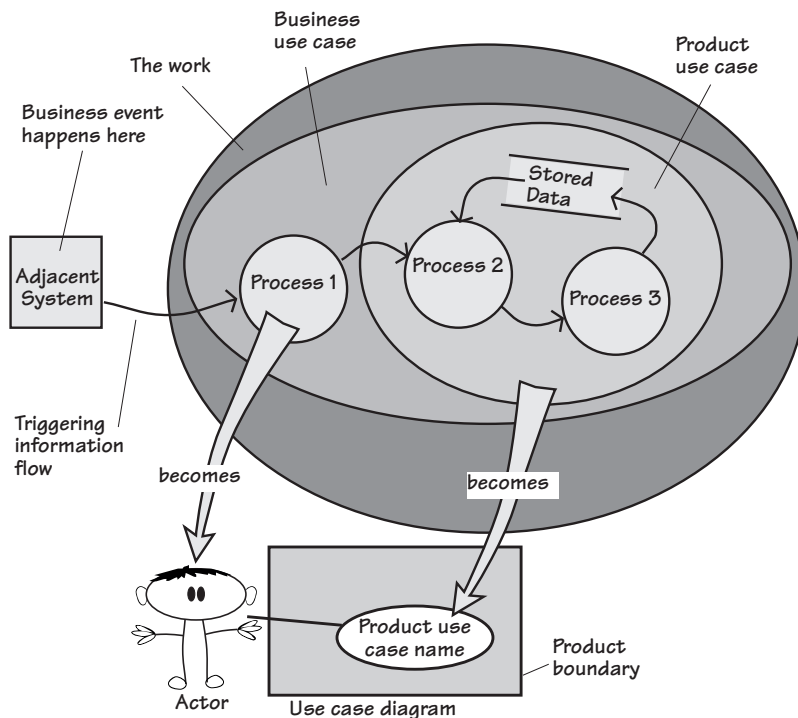


Figure 4.11

The business event is some happening in the adjacent system. The resulting information flow notifies the work of the event and triggers a response (the *business use case*). After study, the requirements analysts and the interested stakeholders decide how much of the business use case is to be handled by the proposed product (the *product use case*). Whatever is immediately outside the scope of the product becomes the *actor*, who manipulates the functionality of the product use case within the product. A UML use case diagram is shown for comparison.

Sometimes, for technical reasons, you might choose to implement a business use case with a number of PUCs. Perhaps you wish to subdivide the work inside the computer into smaller pieces, or perhaps you have the opportunity to reuse product use cases that have previously been developed for other parts of the product or for other products, or maybe different types of stakeholders are concerned with only part of the business use case.

The selection of product use cases is somewhat driven by technical considerations. However, if the product is to be recognizable and usable by its intended users, then its PUCs must be based on the original business events and must be traceable back to the business use case.

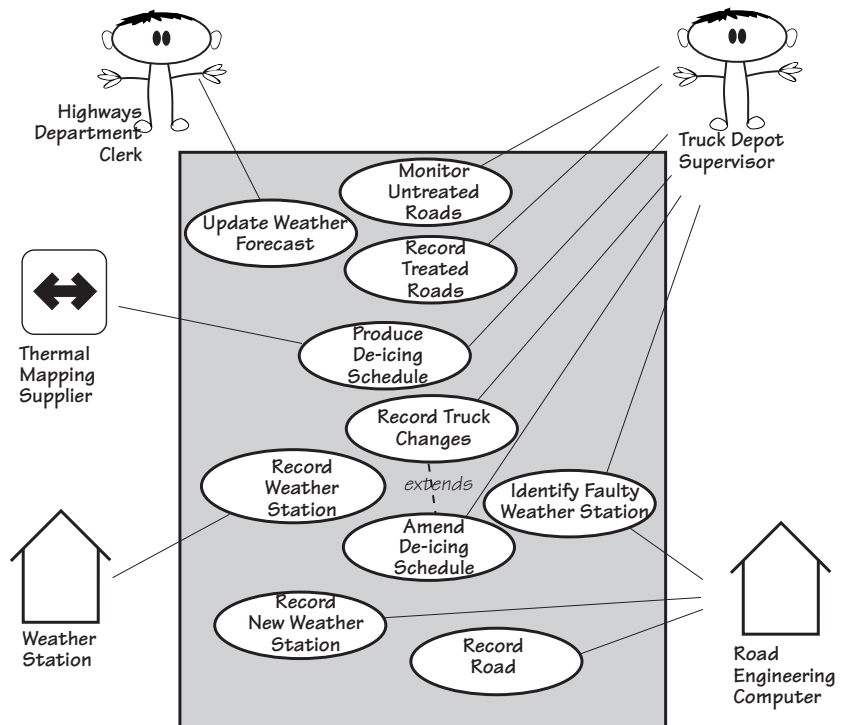
### Actors

When you determine the product use cases, you are also selecting the *actors* who interact with the product.

Actors are the people or systems that interact with the automated product. In some cases, they are adjacent systems that are outside the work—for example, the organization’s customers. In other cases, you appoint actors from inside the organization. Figure 4.12 shows the product use cases that were selected for the IceBreaker product as well as the actors that operate each of the product use cases.

**Figure 4.12**

The product use case diagram for the IceBreaker product, showing the product use cases, the actors involved in each product use case, and the product’s boundary. The different notation used for the actors indicates the way they interact with the product. (These distinctions are explained in Chapter 8, where we look at starting the product.)



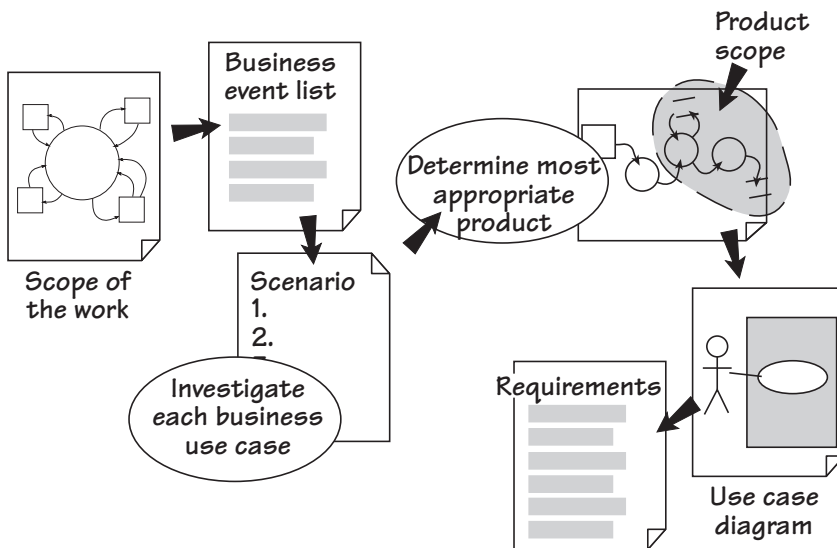
## Summary

Business events and business use cases allow you to carve out a cohesive piece of the work for further modeling and study. By understanding the work being done by each of the BUCs, you come to understand the optimal product you can build to support that work.

If you are outsourcing, you might not determine the product use cases, but work instead on the business use cases. These business use cases can then serve as your negotiating document when you ask your outsourcer which parts of them he can deliver as product use cases.

By using business events to partition the work, you take an external view of the work. You are not relying on how it happens to be partitioned internally at the moment, or on someone's idea of how it might be partitioned in the future. Instead, you partition the work according to the most important view of the work: how the outside world (often your customers) sees it. Figure 4.13 is an overview of this kind of partitioning.

The idea of deriving the product use cases from the business use case means your requirements are grouped according to how the work responds to the business event. The result is a natural partitioning of the work, which results eventually in a product that is more responsive to the real demands of the outside world, and thereby optimally valuable to its owner.



**Figure 4.13**


The flow of requirements discovery: The response to each business event—the business use case—is examined and an appropriate product determined. The analysts write the requirements for each product use case.

*This page intentionally left blank*

# Investigating the Work

# 5

*in which we come to an understanding of  
what the business is doing, and start to think  
about what it might like to do*



The owner's work, for better or for worse, is the starting point of the requirements for any new product you build. Your product is intended to improve the existing work or enable some new capabilities for it. The work might be a small, simple task done by an individual, or it might involve many people and software systems and hardware systems, and make up a significant and critical part of the organization.

Whatever the work, it seems sensible to have a fair understanding of it before attempting any changes. If you charge in and make "improvements" with little or no understanding of what you are changing, then you should hardly be surprised if things do not turn out as they should. In contrast, a little time invested in learning the work will significantly enhance your chances of successfully making a beneficial impact on it. And while you are coming to an understanding of the existing work, you are certain to generate ideas on how to make it better.

In this chapter we discuss how to investigate a piece of work prior to changing it. Normally your change would mean building a software product or some other device to do some or all of the work. This seems a reasonable approach given that when you build a piece of software, you are, in essence, automating a task that if you had enough time, you could do using human labor.

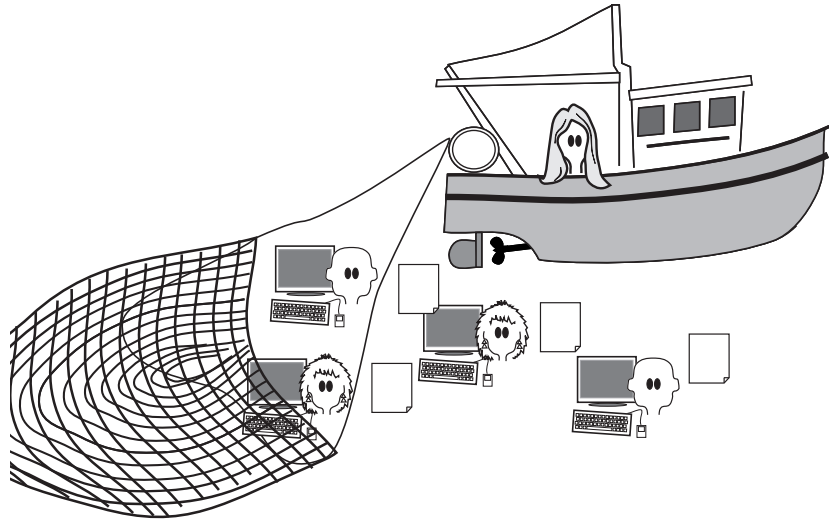
## Trawling the Business

We use the term *trawling* to describe the activity of investigating the business. This term evokes the nature of what we are doing here: fishing. Not idly dangling a line while hoping a fish might come by, but rather methodically



**Figure 5.1**

The business analyst trawls for knowledge by investigating the client's work. The analogy of running a net through the organization is appropriate: You need to sift through much of the business before you can find the best way to improve it.



running a net through the business to catch every possible requirement (see Figure 5.1). With a little experience and good techniques, the skipper of a trawler knows where to fish so that he gets the fish he wants, and avoids the ones that he doesn't. The objective in this chapter is to show you some trawling techniques, and give you guidelines on how to get the best from them.

While it is important to uncover the current business, including both its data and its processes, it is also important that you do not spend too much time doing so. Keep in mind that this is the beginning of the analysis process, and you would like to get through this step as rapidly as possible. Also keep in mind that the business you are studying is about to be changed. Given these caveats, we suggest that you study the current business as quickly as possible—you can always come back later to get additional information should it be necessary.

This chapter explores the techniques for discovering the business processes, and the people involved in those processes. Inevitably, you will need a variety of techniques to achieve this feat, and you will find that not all techniques are equally acceptable to your stakeholders. For this reason, it will be necessary to vary your approach to best suit the stakeholders who are providing the requirements.

When you are reading about the trawling techniques, you will see our suggestions on when and how they are used. Use these suggestions to determine for yourself what is most appropriate for your stakeholders, your preferred way of working, and your project. We have included an owl to provide guidance on whether the technique is applicable to your situation.

---

*While it is important to uncover the current business, including both its data and its processes, it is also important that you do not spend too much time doing so.*

---

## Formality Guide

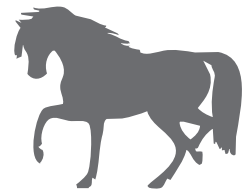
The study of the current business is not intended to be a lengthy process. You are encouraged to do it as quickly as possible, and to stop doing it once you have enough information about the current state of the business to enable you and your stakeholders to move on to the next stage. Also keep in mind that requirements are not solutions; you have to learn the requirement before you can find the solution. It just doesn't work the other way round.

Rabbit projects need to understand the work as it currently is, as well as what the work is to be. Because rabbit projects are usually iterative, you are likely to visit the current work in small slices and then proceed to find its essence and ultimately its solution. This cycle is repeated until the product is complete. This iterative way of working should minimize any documentation of the current work. However, that does not obviate the need to *understand* the work.



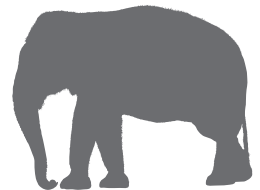
Some of you will be working on an agile team with a product owner or customer representative. We have found (unfortunately) that a single person is unlikely to have a good enough understanding of the wider business to be able to provide all of the needed information. We suggest you adopt some of the trawling techniques to enhance your iterative team's understanding of the work, and as a result enhance the product you are building.

Horse projects, due to their larger number of stakeholders, probably make more extensive use of apprenticing, interviewing, and use case workshops. These techniques generate some documentation, and while documenting the current work is by no means the objective of the project, it is extremely useful as input to subsequent decisions.



Horse projects are more likely to be dealing with critical infrastructure systems, so knowledge of the work and its goals is important to the project.

Elephant projects, due to their larger number of stakeholders, need to document their findings as they go about trawling for their requirements. Because of the more extensive set of possibilities for elephant projects, we suggest that you read this entire chapter, paying special attention to the "owl recommendations."



Projects using outsourcing are always elephants—they need a formal specification. Additionally, the artifacts produced by investigating the work should be retained for future maintenance.

## Trawl for Knowledge

We build products to help us do work. For our purposes, it doesn't matter whether the work is processing insurance claims, analyzing blood samples, designing automotive parts, predicting when ice will form on roads, keeping track of a "things to do" list, controlling a telephone network, downloading

music or movies, monitoring a household, manipulating photographs, or one of many other human activities. In all instances, the product that you are being asked to build must improve this work.

Figure 5.2 shows the Volere process and highlights the part where you investigate the work with a view toward discovering the best product to enable or improve that work.

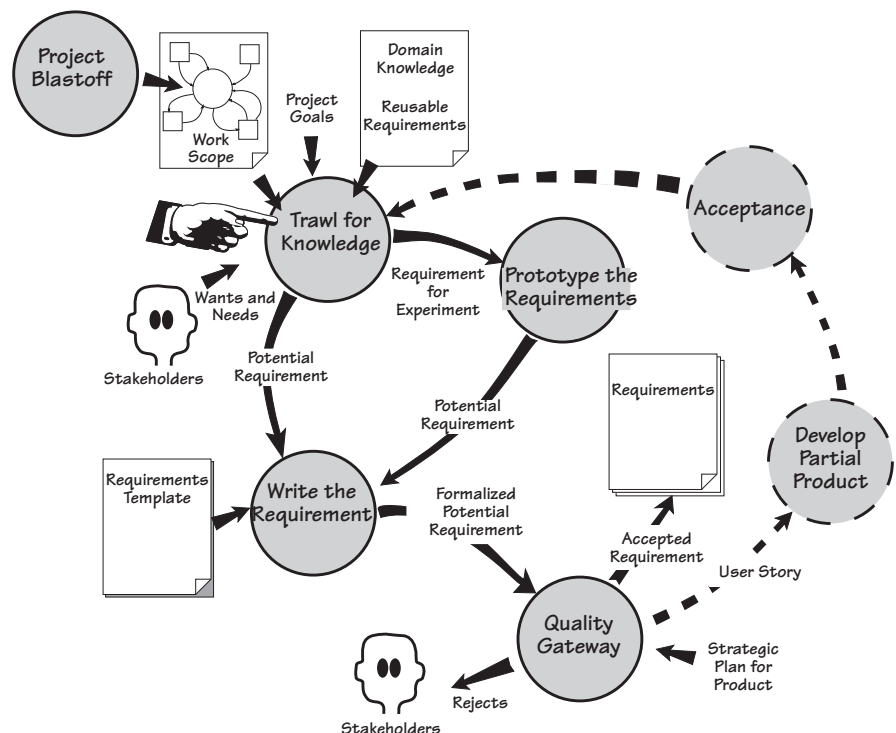
When you are trawling for knowledge, the first task is to investigate and understand the work as it is currently being done. It is not always necessarily to document this information, but you must certainly understand it. Once you have a fair understanding of the current work, then you can derive its essence. That is, you can strip away the current technology to get a clean picture of the real business.

The essential view is then developed in conjunction with the stakeholders to arrive at the requirements for the new product. This is not a laborious process—just a thorough one. We will look at it over the next few chapters.

The trawling activity uses outputs from the project blastoff—the scope of the work, the goals of the project, and the constraints that apply to any solution. The blastoff also identified the stakeholders involved in the project and the potential users. These stakeholders are the people with whom you consult to get an understanding of the work.

Figure 5.2

The trawling activity is central to the requirements process. It uses the outputs of the project blastoff activity as its starting point for investigating the work and accumulating knowledge about it. Over the next few chapters, we will develop this knowledge into requirements for the product to be built. The dotted lines on the diagram indicate how trawling works when you are using iterative development.



## The Business Analyst

The business analyst is also known as a systems analyst, a requirements engineer, and probably several other titles. We use “business analyst” because it is the most commonly used name. Whatever name you use, this person is an investigator and a translator: He has to inspect the work, interview the business stakeholders, understand what they are saying, and then translate that knowledge into a form that can be communicated to and understood by the developers. Initially, the business analyst’s task is to record, clarify, and question the current state of the business. Later, as the analysis progresses, the emphasis changes from recording an existing system to thinking about a new one. For the moment, the task of the business analyst is this:

- *Observe and learn the work, and understand it from the point of view of the owner.* As you work with your users, you study their work and question them about what they are doing, and why they are doing it.
- *Interpret the work.* A user’s description of some work is not always factual despite the user being the expert on that part of the work. The analyst must filter the description to strip away the current technology, thereby revealing the underlying essence of the work, not its incarnation.
- *Record the results in the form of stakeholder-understandable analysis models.* The analyst must ensure that he and the stakeholders have the same, and agreed, understanding of the work. We suggest using models as a common language for communicating your knowledge to the stakeholders.

*There is a lot to know about trawling—and we know that not all trawling techniques are applicable to all projects. The owl gives you guidance on whether the technique is applicable to your situation. By all means, read this entire chapter—but realize that you don’t need to use all of the trawling techniques on every project.*



Many techniques are available to help with the task of studying the business. We provide you with several choices here because we know that no single technique works in every situation. Instead, you have to select the technique that works best for you at the time. In our discussions of the techniques, we indicate when and why one would be useful. But don’t just take our word for it—try to connect each technique to your own situation, and consider where and when each would be most advantageous.

Consider your stakeholders. They are *conscious* of some of their processes and requirements and bring them up early. At the same time, they are *unconscious* of others—things that are so ingrained into the stakeholders’ work that they have forgotten they exist, or forgotten how they are done. The techniques that capture the conscious processes may not necessarily work for the unconscious ones. Then there are the *undreamed-of* processes—those functions and features that the stakeholders are not aware they could have.

Undreamed-of needs exist because the stakeholders do not realize they are possible, perhaps because the stakeholders lack sufficient technological sophistication, or perhaps because they have never seen their work in the way that you will show it to them.

Whatever the situation, your responsibility is to bring the conscious, unconscious, and undreamed-of work to the surface.

## Trawling and Business Use Cases



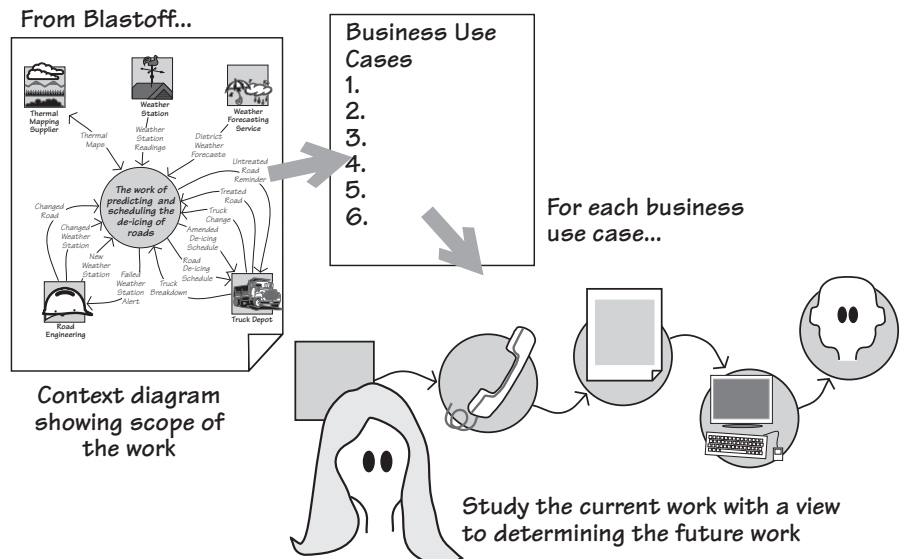
*Business use case are so fundamental to the requirements activity that we urge you, whatever your situation or project type, to consider doing your requirements trawling one business use case at a time. If you have not already read Chapter 4, Business Use Cases, we encourage you to do so now.*

From the work context diagram, you determine the business events and the resulting business use cases. In Chapter 4, we discussed business events, exploring how happenings outside the work cause a response inside the work. This response is a *business use case*, and we suggest that you do your work by studying one business use case at a time.

The business use case is the functionality that the work does in response to a business event. When the triggering data flow enters the work, the work starts to process it. If the triggering data flow arrives via a telephone call, then a person might be there to answer it. Alternatively, an automated telephone system might prompt the caller to identify the nature of the call. In the end, it doesn't matter *how* it is done; the important thing is *what* is done. This functionality is what you study when trawling, as is illustrated in Figure 5.3.

Figure 5.3

Business events are determined using the triggering data flows from the adjacent systems on the context diagram. The business use cases are the work's responses to the business events, and these are studied until the analyst understands the way the work functions.



## The Brown Cow Model

When you are investigating the work area, there are a number of ways in which you can look at it—*viewpoints*—and there are four really useful ones. We shall demonstrate them using a brown cow.

The Brown Cow Model,<sup>1</sup> shown in Figure 5.4, takes four views of the work; the two axes separate *What* from *How*, and the *Now* from the *Future*.

Let's start in the lower-left quadrant of the model: How-Now. This is sometimes—but not always—the place to start. How-Now shows the implementation of the work as it currently exists, including the physical artifacts, people, and processors used to do the work. You use this view when you need to get enough of a grounding to begin to ask other questions.

When you set out to build some new product, you are not trying to simply duplicate whatever you have at the moment; no gain is realized in doing that. Instead, by eliminating the technological fossils and obsolete organizational procedures from the current situation, you begin to see the pure, unadulterated business problem.

This brings us above the line to What-Now. This abstract view shows the real business policy, or as we prefer to call it, the *essence* of the work. This view is completely technologically neutral, and it shows the business as if no machines, people, or organizational departments existed. We use this view to cleanse our ideas on what the current business is actually doing without inhibiting it by referring to processors and physical artifacts that might not be part of a future implementation.

Moving to the upper-right quadrant, we get to the Future-What view. This view shows the business as your owner wants to have it, but still without the

### FOOTNOTE 1

If you are wondering, the name of this model comes from a piece of English elocution. Students being taught to speak correctly with rounded vowels have to clearly enunciate, "How now brown cow?" The first segment of this model is the How-Now segment—hence the Brown Cow name for the model.

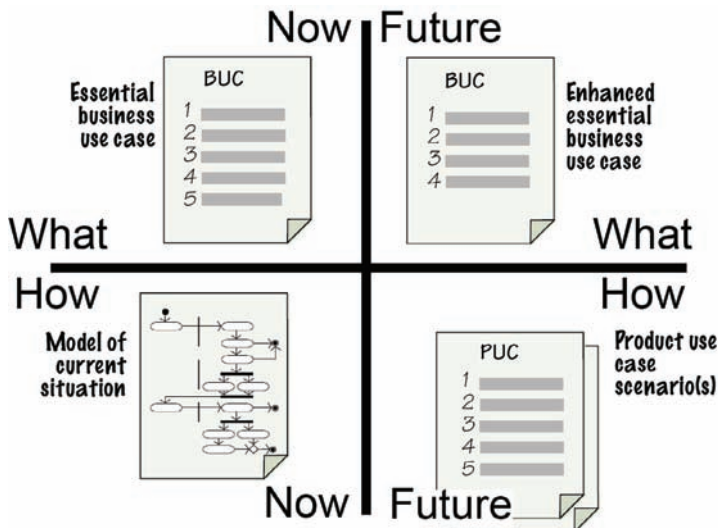


Figure 5.4

The Brown Cow Model. This shows four views of the work, each of which provides the business analyst and the stakeholders with information that is useful at different stages of the requirements discovery process.

technology that might be used to implement that business. It is the purest version of the proposed future state of the business area. The value of this viewpoint is that it shows—so you can discuss with your stakeholders—precisely what the owner would like to do, without worrying about how the technology might do it.

The last quadrant, the lower right, is Future-How. Here you take your idealized view of the future business policy, and augment it with the technology and people needed to bring it into the real world.

It is generally not sufficient to simply have two views of the business—the “as is” and the “to be.” These views never get away from the implementation, never see the pure business problem in its true light, and as such are unsuitable for any kind of serious innovation or system development.

## The Current Way of Doing Things (How-Now)



*Building models as a way of investigating the work is best done when you need to understand a medium-to-large work area for which no documentation exists. This technique is also used when the current users struggle to give you an idea of how the work fits together. It is also useful when you know that there will be a significant legacy from the existing work.*

We have mentioned several times—and probably will do so again several times more—the need to understand the work. This is best done by being actively involved in investigating the work rather than by being a passive onlooker.

Despite any bad reputation the current work may have, it is still useful: It contains functionality that is making a positive contribution to the business. Naturally, much of this functionality must be included in any future system. You may implement it differently with new technology, but its underlying business policy will remain almost unchanged. Thus one reason to build a model of the current work is to identify which parts you need to keep.

You can use models to help you understand the work but, paradoxically, you cannot build a model without understanding the work. This paradox follows from the way the modeling activity leads you to ask all the right questions. Any useful model is a *working* model—it is functionally correct in that you can demonstrate the model’s outputs are derivable from its inputs. As a consequence, if your model isn’t working, it means you simply haven’t asked enough questions to get enough right answers (see Figure 5.5). Alternatively, we could say that as the model develops, it becomes increasingly more obvious what you don’t know, how much you don’t know, and what the business people don’t know.



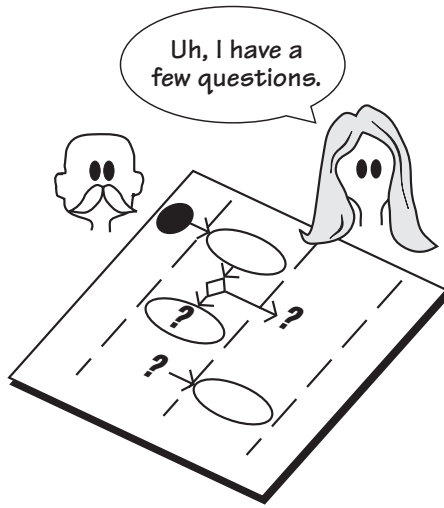


Figure 5.5

Knowing where to ask questions, and which questions to ask, is almost as useful as knowing the answers. If your model has flows going nowhere, processes without inputs, read-only or write-only data stores, or other breaches of the modeling conventions, then you need to ask your stakeholders more questions about those aspects of the work.

Your models record the work and demonstrate your understanding of it. Because the model is a common language between you and your stakeholders, you can come to have the identical understanding of the situation.

When you model the current work, keep in mind that you are not attempting to specify a new product, but rather merely establishing the work that the users currently do. Most likely the users will describe their work in a way that includes the mechanisms and technology they use to get the work done. These mechanisms are not requirements for the new system—you must look beyond them to see the underlying policy of the users' system. That understanding emerges when you move "above the line" and work with the two segments of the Brown Cow Model above the horizontal axis that deal with *what*, rather than *how*. We shall get to this step in a little while.

Modeling the current system should be done as quickly as possible. Figure 5.6 shows a model of an existing system, built by the authors in conjunction with staff at City University, London. This model was built in about 15 minutes—but we were drinking coffee at the time.

Restrict the amount of detail you include in your models of the current system—there is little point in modeling every tiny facet of something you are about to replace. The ideal model contains enough detail to give you an understanding of the work, and no more. The detail shown in Figures 5.6 and 5.7 is about right. That is, these models show the major parts of the current situation. Such a model would allow the stakeholders to verify that it is a good-enough representation of the work as it stands, and it gives the requirements analyst places to make further inquiries.

#### READING

Robertson, James, and Suzanne Robertson. *Complete Systems Analysis: The Workbook, the Textbook, the Answers*. Dorset House, 1998. This book demonstrates how to build models of current, future, and imaginary work.

***The current work contains many functions that contribute to the continuation of the business. Naturally, these functions must be included in any future work.***





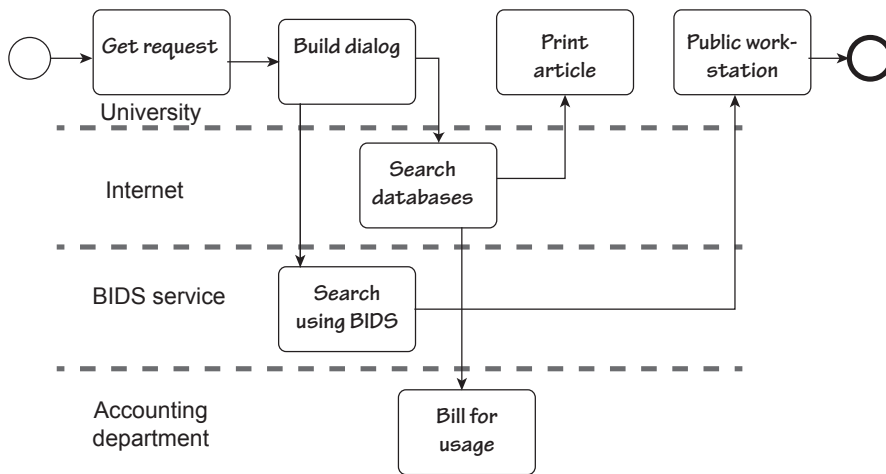


Figure 5.7

A UML activity diagram showing the same piece of work as illustrated in Figure 5.6. Business analysts should use whatever models they feel most comfortable with.

One aspect of the model that should not, within reason, be restricted is the area of the work covered by the model. Here it is almost—and we stress “almost”—a case of “the more, the merrier.” Your models should cover all the work that could possibly be relevant to your product, those parts of the business that could contribute to the new product, and those parts where operational problems have popped up in the past. The other areas worth covering are those where the business is not well understood.

The point of having a large scope for models of the current work is that requirements analysis is really *work reengineering*. You specify products to improve work; thus, the more of the work that you study, the more opportunities to improve it that will emerge. The greater the scope of your study, the better your understanding and the better chance you and your stakeholders have of finding areas that will benefit from improvement.

The current model is also used to confirm the work scope. During the blastoff exercise (Chapter 3), you and the stakeholders built a context model to show the scope of the work you intended to study. Any models you build while looking at the current situation should confirm that all of the appropriate parts of the work are included in the context. If at this stage you discover areas that would benefit from your attention, parts of the work that need to be understood, or anything at all that should be included, then now is the time to adjust the context. Such a revision does entail getting the agreement of the stakeholders, but it is better to enlarge the scope now than to end up with a product that lacks important functionality.

## Apprenticing



*Apprenticing is particularly useful for in-house work. The underlying assumption for apprenticing is that users are currently doing work, and you, as the requirements analyst, have to understand their work. This work could be clerical, commercial, graphic arts, engineering, or almost anything short of brain surgery.*

*If significant parts of the current work and systems are likely to be reimplemented, then apprenticing is appropriate. Please keep in mind, however, that you will not reimplement the work exactly as is. We recommend that all apprentices refer to the section on essence.*

Apprenticing—based on the old idea of masters and apprentices—is a wonderful way to observe and learn the work as it really happens. In this case, the requirements analyst is the apprentice, and the user assumes the role of master craftsman. The analyst sits with the user at the user's place of work, learning the job by making observations, asking questions, and perhaps doing some of the work under supervision. This technique is also sometimes known as “job shadowing.”

It is unlikely that many users can explain what they do in enough detail for the business analyst to completely understand the work and thereby capture all the requirements—if you are away from your work, you tend to generalize. Generalizations can be useful, but they do not provide enough detail for them to work every time.

Nor can you expect users to have the presentation and teaching skills needed to present their work effectively to others. Conversely, almost everyone is good at explaining what they are doing while they are doing it.

If the user is doing his job in his normal workplace, he can provide a running commentary and provide details that might otherwise be lost. So if you want to get an accurate explanation of the work, go to the work; sit beside the user in the normal workplace and get a running commentary on the work as it happens. While he is working, the user can describe his task precisely and tell you why he is doing things. If the explanation is not clear, the apprentice asks questions: “Why did you do that?” “What does this mean?” “How often does this happen?” “What happens if this piece of information is not here?” You also get to see the things that can go wrong, and the special cases, and the work-arounds he uses when things are not normal. Through the process of apprenticing, you come to see all the cases and the actions the user takes for each.

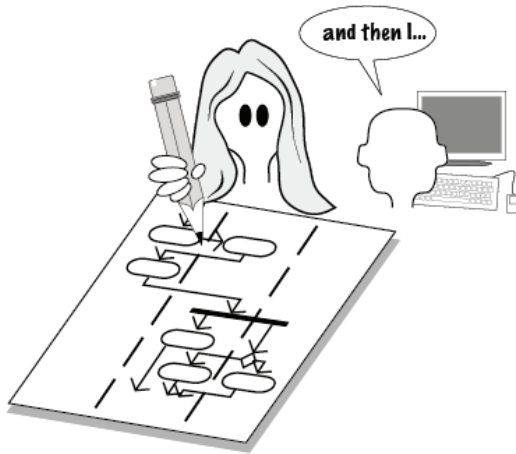
Apprenticeship can be combined with modeling. As you observe the work and the user explains it, you sketch a model of the tasks and their connections with the other tasks (see Figure 5.8). As you build your models, feed them back to the user and confirm that they are correct. Naturally you use this feedback to raise questions about any areas of uncertainty.

### READING

Holtzblatt, Karen, Jessamyn Burns Wendell, and Shelley Wood. *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design*. Morgan Kaufmann, 2004.

“Nobody can talk better about what they do and why they do it than they can while in the middle of doing it.”

—Hugh Beyer and Karen Holtzblatt. *Contextual Design*

**Figure 5.8**

The requirements analyst learns the work while sitting at the user's desk, sometimes building models of the work while learning it.

When you are apprenticing, you are an interpreter as well as an observer. When you are looking at the current work, you must abstract away from what you see to overcome the user's close connection to the physical incarnation of the work. In other words, the artifacts, the technology, and other inputs that are currently used must be seen as a product of a previous designer. Someone, some time ago, decided that was the best way to do the work. But times have changed. Today it may be possible to do things that were impossible yesterday. Better ways may now be available—ways that take advantage of up-to-date technology, that use streamlined processes, that simplify the work or automate some or all of it.

But first you have to abstract. While you are learning the work by seeing it done in real-world conditions, you are abstracting from the current technology to find the underlying essence of the work. We will come back and have a good look at essence a little later.

## Business Use Case Workshops

*BUC workshops are useful to most projects, and are the most commonly used requirements technique. These workshops look at one slice of the business with the objective of finding the ideal work. You must overcome geographical limitations and ensure that you can assemble the right combination of specialized stakeholders who have an interest in the business use case. These workshops are particularly useful when you are making fundamental changes to the work.*



In Chapter 4, Business Use Cases, we discussed business events and how they trigger a response by the work. We called this response a business use case (BUC). We hope that by devoting an entire chapter to business events and business use cases, we have unequivocally signaled the importance of partitioning the work according to how it responds to the world outside it.

Refer to Chapter 4, Business Use Cases, for a complete explanation of business events and BUCs.

**READING**

Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley, 2002. Gottesdiener's book covers in detail how to plan and conduct requirements workshops.

Hass, Kathleen, and Alice Zavala. *The Art and Power of Facilitation: Running Powerful Meetings (Business Analysis Essential Library)*. Management Concepts, 2007.

See Chapter 3, *Scoping the Business Problem*, for more on how to discover the relevant interested stakeholders.

Now that you have partitioned the work, the business use case workshop is an effective way of understanding each partition and making improvements. The workshop is where interested stakeholders describe or reenact the work they currently do, and discuss the work they desire to do, for one specific business use case. Your task is to record this work in a way that the stakeholders can understand it and agree that it is an accurate portrayal. As a general rule, we suggest that you use scenarios to show the functionality of the business use case (see Figure 5.9). Later, you will use this recording of the improved work to derive the requirements for the product to support this work.

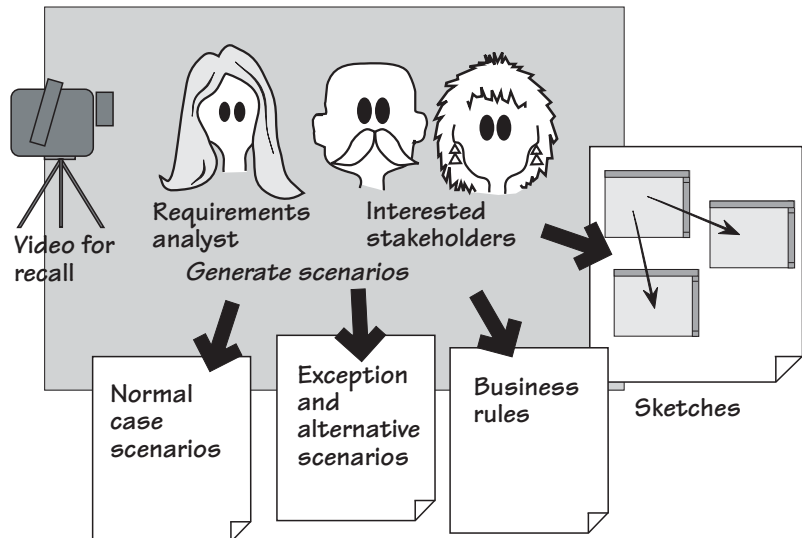
In Chapter 6, we talk about scenarios as a way of telling the story of the business use case in a fairly structured way. We propose that you use scenarios as part of the business use case workshop—they are easy to understand, and provide a convenient focal point for the workshop. Feel free to peek into Chapter 6, but for the moment it will suffice if you think of a scenario as a tool for breaking the business use case's functionality into a series of steps.

The analysts and the stakeholders work together on a business use case and record the following information:

- The desired outcome for the BUC
- A normal case scenario that describes the work done by the BUC
- Exception scenarios describing what can go wrong and what the work does to correct them (these can be postponed until later if desired)
- The business rules applicable to the business use case
- Sketched prototypes used to help stakeholders visualize the business use case—these throwaway sketches are optional and are not intended to be kept beyond the requirements phase

**Figure 5.9**

The business use case workshop records the proposed functionality using scenarios and sketched prototypes. The workshop serves as a forum for interested stakeholders to communicate effectively, express their understanding, ask questions, and give their aspirations for the work.



## Outcome

The outcome is what the organization hopes to achieve from this business use case, cast in terms of results, not outputs. For example, suppose the business use case is “rent a car to a customer.” The desired *outcome* is that the customer drives away in the car of his choice, the rate selected is equitable, the details are recorded, and the transaction is completed with the minimum inconvenience to the customer at minimal cost. The *outputs* of the same business event are the rental document and some recorded data. Achieving the output does not necessarily guarantee the outcome.

---

*Think of outcomes,  
not outputs.*

---

Note that the outcome is a business objective, not a way of achieving something. Outcomes are expressed from the point of view of the owning organization—what it wants to achieve. The outcome gives your business use case its reason for existing.

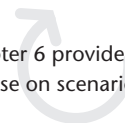
The outcome should be able to be written as one sentence along the lines of “When this business event happens, this is what we want to achieve.”

## Scenarios

Scenarios describe the work being done by the business use case as a series (usually between three and ten) of steps. The scenario is a stakeholder-friendly document that serves as the focal point of the workshop. The steps do not have to be detailed; the intention is not to capture every small part of the business use case, but rather to obtain a business-level picture of the work that stakeholders can use to arrive at a consensus.

The normal-case scenario shows the actions of the business use case if everything works as desired and no mistakes are made, no wrong actions are taken, or no other untoward happening occurs. The intention is for you and your stakeholders to reach consensus on what *should* happen. Other scenarios focus on the exception cases—when unwanted things go wrong—and the alternative scenarios—when the users of the business use case may take some allowable variations on the normal case.

The collection of scenarios represents the interested stakeholders’ consensus on what the work should do in response to the business event. We stress again the importance of understanding the work, and not just the product.



Chapter 6 provides a full treatise on scenarios.

## Business Rules

The business rules are management prescriptions that transcend and guide the day-to-day business decisions. For example, we discovered these two business rules in the road de-icing project:

The maximum length of a truck driver’s shift is 5 hours.

The engineers maintain the weather stations once a week.

These rules come to the surface as the stakeholders discuss the work for each business event. Later they are used to guide the functional requirements and to help to discover the meaning of stored data. Naturally, any product you build must conform to the business rules, and often it is the product that has to enforce them.

There is no set form for the business rules. In most cases you will be able to find, with the help of the interested stakeholders, existing documents that spell out these rules. The rules must, of course, become incorporated into the business use case scenarios and the subsequent requirements.

The business use case workshop gives the interested stakeholders the opportunity to bring out all the relevant business rules.

## Interviewing the Stakeholders



*Interviewing is used by almost all projects, as it is really a part of all elicitation techniques. Despite its omnipresence, we suggest that you do not use interviewing as your only technique for gathering requirements. Rather, you should conduct interviews in conjunction with the use of other techniques discussed in this chapter.*

Interviewing the stakeholders is a technique that is commonly employed, but is not without its problems. The interviewer relies on the interviewee to know—and be able to verbalize—all the necessary knowledge about the work. This method may work very well when people are familiar with the work, but that knowledge is often restricted to their own immediate area. There are likely to be few people in the organization who understand enough of the business to be the sole person to describe it for you. Interviewing also requires some abstraction and communication ability on the part of the interviewee. For this reason, it is wise to avoid relying on interviews as your sole method of gathering requirements; instead, use them in conjunction with other techniques.

That cautionary note notwithstanding, interviewing skills are highly useful in other contexts. For example, a requirements analyst can “interview” a model or a document. The skill here is to know which questions to ask of the model, or more realistically, the appropriate stakeholder for the model.

Some requirements analysts draw up a questionnaire and send it to the stakeholder in advance of the interview. While this preparatory step gives some structure to the subsequent interview, we have found few stakeholders who are motivated enough, or who have time enough, to fill in a questionnaire prior to meeting the analyst. We suggest that you send a brief agenda listing the topics that you wish to cover in your interview along with the planned duration of the interview. This notification at least gives the interviewee a chance to think in background mode, to have needed material close by, or to ask subject-matter experts to be present.

Stakeholders should not remain completely passive during the interview. Instead, do your best to involve them by building models—business event responses, use cases, scenarios, and so on—during the interview. This approach creates a feedback loop between you and your stakeholders, and it means you can iteratively test the accuracy of what you are being told. Follow these guidelines to make your interviews more effective:

- Set the interview in context. This step is necessary to avoid having your stakeholders talk about something irrelevant to your purpose. It also gives them a chance to withdraw gracefully if they have not prepared for the interview.
- Limit the duration of the interview to the time stated in the agenda. We find that interviews running for more than an hour (90 minutes tops) tend to lose focus.
- Have business use cases serve as an anchor for the interview. Users recognize business use cases (although they may not necessarily call them by that name), and it makes for more directed conversations if you talk about their work one business use case at a time.
- Ask a question (more on this in a moment), listen to the answer, and then feed back your understanding.
- Draw models and encourage the user to change them. Plenty of models (e.g., data flow diagrams, activity diagrams, sequence diagrams) are available to help you communicate your understanding of a process. You can also construct data models for information, and mind maps to link different subjects.
- Use the stakeholders' terminology and artifacts, both conceptual and real. If the stakeholders do not use their own language, then you force them to make technological translations into terms they think you will understand. This, sadly, usually leads to misunderstandings. By contrast, if you are forced to ask questions about their terminology, you inevitably make new discoveries.
- Keep samples or copies of artifacts and log them for future reference. Artifacts are the things the stakeholders use in their daily work. They can be real things: documents, computers, meters, spreadsheets, machines, pieces of software. They can also be conceptual things: status, contracts, schedules, orders. Artifacts will inevitably cause you to raise questions when you examine them later.
- Thank the stakeholders for their time and tell them what you learned and why it was valuable. After all, they have lots of other things to do. Talking to you is not the reason they are employed, and they often view the interview as an interruption.

---

*Feed back your understanding. Build models while you are interviewing the stakeholder.*

---



---

*Thank the stakeholders for their time.*

---



- Write down what you were told. We guarantee you'll have more questions.

Note taking is almost a skill set of its own. We encourage the use of mind maps as a note-taking device—we talk about these later in the chapter. You might also consider some of the smartpens on the market. These devices record an electronic version of your notes (easy to share with the rest of the team) and some record the voice part of the interview. Livescribe pens are popular at the time of writing, and some others offer similar technology.

### Asking the Right Questions

We suggested earlier that the business use case was the ideal unit of work to interview someone about. We also urge you to use the Brown Cow viewpoints (shown in Figure 5.4) as triggers for asking the right questions. Let's look at this process by using an example from the IceBreaker road de-icing project.

Suppose that you have an interview with the Chief Engineer. The business event that you want to explore is *Road Engineering Installs New Weather Station*. Your questions should focus on what the work does to respond to this business event.

Business Analyst: "What happens when your engineers install a new weather station?"

Chief Engineer: "The engineers let us know where the new weather station is located, and we keep a record of it."

BA: "How do you keep a record of the new weather station?"

CE: "I'll show you the spreadsheet that we use; here it is."

BA: "Thank you. I'd like a copy of this weather station spreadsheet. I see that you keep track of the weather station number, geographical coordinates, and installation date. What else do you keep track of?"

CE: "We have another computer system where we have the maintenance history for each weather station. There are some double entries, but that's the way it is."

BA: "Are there any other facts about the weather station that would be useful to you?"

CE: "Well, we would like to know the manufacturer and performance of each weather station."

BA: "What do you mean by performance?"

CE: "Which weather stations need the most maintenance. This would help in planning which new technology we spend our budget on. However we've never been able to do that in the past."

---

*Note that the analyst begins with a "W" word—what, why, where, who—to open up the question and elicit information.*

---

BA: “Would it be acceptable to you if you had a new system that integrates all the information about the weather stations, their maintenance, and their performance?”

CE: “What, you mean all on one screen?”

BA: “Yes, something that makes it possible for you to track a weather station and its lifetime performance.”

CE: “That sounds marvelous, but we’d have to check it with some of the engineers.”

BA: “Well, once I’ve made sure that I have all the facts about the weather stations properly defined, then I could do a quick mockup to show the engineers our ideas. How would that be?”

CE: “I like the idea.”

BA: “Thanks for your time. I’ll go back and review my notes and the samples you have given me, and I’ll send you a brief summary of this interview. I might need to phone you with a couple of questions. Then I’ll make an appointment to explore the mockup with the engineers. Is this approach all right with you?”

CE: “Yes it is. Thank you.”

In this interview, the business analyst uses the Brown Cow viewpoints to help explore the business use case. He starts by asking how the work is done now (How-Now) and discovers the spreadsheet and the other computer system for maintenance of the weather stations. He asks, “What else do you keep track of?” to discover the attributes that are recorded (What-Now) when a new weather station is installed. The question about “What other facts” is designed to uncover other facts that are not currently recorded or made use of (Future-What). And raising the idea of an integrated system and a mockup (Future-How) is signaling the chief engineer that there will probably be a better way of doing things once the business analyst has properly understood the business of installing weather stations.

Notice that the business analyst used a lot of *open questions*. These begin with “What,” “How,” “When,” “Where,” “Why,” or “Who,” and they encourage the interviewee to give a detailed answer rather than a “Yes” or “No” answer.

---

*The Brown Cow viewpoints help you explore the business use case as it is now, its essence, and what it might be in the future.*

---

## Listening to the Answers

Being a good listener means being able to understand someone else’s meaning behind the words that they are using. This is difficult to do because we all have our own view of the meaning of words that is based on our own assumptions, experience, and preoccupations. Also, especially when interviewing someone, we are often afraid of silence and might think, “If I’m not saying anything, then maybe the interviewee will think that I don’t know



### READING

The following are books that provide insights into how to ask better questions and listen to the answers.

O'Connor, Joseph, and John Seymour. *Introducing Neuro-Linguistic Programming*. Conari Press, 2011.

Sullivan, Wendy, and Judy Rees. *Clean Language: Revealing Metaphors and Opening Minds*. Crown House Publishing, 2008.

Weinberg, Jerry. *Quality Software Management. Volume 1: Systems Thinking. Volume 2: First-Order Measurement. Volume 3: Congruent Action. Volume 4: Anticipating Change*. Dorset House, 1992–1997.

what I'm doing." In fact, the reverse is true: It is comforting to have some silence and time for reflection. If you are the interviewer, however, even a few seconds silence seems like a lifetime.

If you want to improve your listening skills, you can learn a lot from the fields of psychology, sociology, and family therapy. These disciplines have realized that to help people with problems, you first have to understand what those people mean by their words and behavior.

In their book *Clean Language*, Wendy Sullivan and Judy Rees talk about the power of "attending exquisitely" to another person's words. One of their hints on improving listening skills is "Put your attention on what the other person is actually saying rather than on the person themselves or what you think they might mean by their words." Another helpful piece of advice: "Repeat back some of their words or phrases exactly as you heard them." By doing so, you demonstrate that you are, in fact, listening to what that person is saying, and this feedback encourages that person to expand further on what he means by his words.

Most of all, if you want to become a better listener, don't talk; develop an appreciation of silence.

## Looking for Reusable Requirements

*Many of our projects deliver similar products. That is, if you work for a finance house, then almost all of your projects will deliver some kind of financial system. Given that your organization has been doing this for a number of years, it is likely that someone, at some time, has studied a piece of work that is similar to yours and has written requirements for a product that is similar to yours. You don't have to do it all again when you can borrow from those who have gone before.*



Investigating the work is a time when you observe and interpret. The interpretation we are looking for here is one where you see a business process and recognize its similarities with other processes. By making abstractions—that is, by ignoring the subject matter and concentrating on the processes (look at the verbs, not the nouns)—you are often able to interpret one piece of work as having the same functionality as another. This does not mean they have the same subject matter, but rather that they have similarities in the way that both pieces of work are done.

As an example of this, one of our clients, the international section of a bank, had 20 different products. The products ranged from letters of credit, to guaranteed foreign bank loans, to guaranteed funds, and so on. At first glance, the users appeared to handle each of these products differently. However, a common pattern emerged as we studied the work—we were looking for similarities, not differences. We observed that each product was, in fact, a different way of guaranteeing that exporters got paid for their goods in

---

**Look for similarities,  
not differences.**

---

foreign countries. The end result was that we were able to borrow heavily from one project to feed the next. We had to change names and make some alterations to processes, but by and large, by using abstracted requirements from other projects, we were able to save significant project time.

We suggest building abstract models of the work structure—that is, models that do not give specific technological names to things, or use distinctive terminology associated with one part of the organization. Such models also do not apply to any particular user, or use terminology identified with a particular user. They are abstracted from their source—they use categorizations rather than specifics, generics rather than actual instances. Instead of modeling the work the way any single user sees it, they model the *class* of work as all users could see it.

This kind of abstraction enables you to discover whether the same work pattern exists in another part of the organization. It has been our experience that although the names and the artifacts may vary, the same work pattern occurs several times in an organization. We have used the recurrence of patterns first to understand the requirements more quickly, and then to deploy the implementation of one part of the work to suit another part.

Chapter 15 deals with reusing requirements. We suggest that you visit this chapter if you suspect you have the opportunity to reuse requirements and other artifacts from previous projects.

See Chapter 15, Reusing Requirements, for more on patterns.

## Quick and Dirty Process Modeling

*Quick and dirty process modeling is a technique for building quick models of business processes to gain understanding and consensus about the current work. We find that business stakeholders relate very well to the dynamic nature of these kinds of models.*

*The technique is useful when much of the legacy system is to be replaced. It can also be used when the stakeholders are geographically dispersed.*

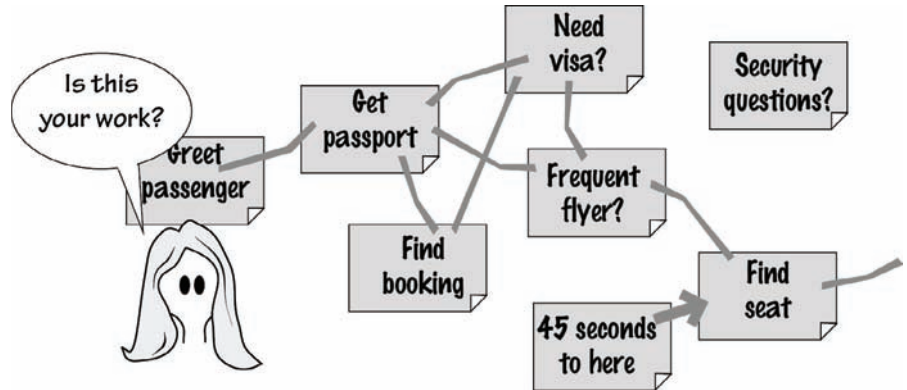


Quick and dirty process models simulate the current reality, and of course can be used to model any future processes. They are not formal models in the sense of UML activity diagrams and the like, but rather are put together using whatever physical artifacts are to hand. We find that Post-it notes are the most convenient way to build these models, but they are not confined to that medium (see Figure 5.10).

The technique is more or less to build a process model using large Post-it notes, or index cards, for each of the activities in the process. You can stick the notes onto a whiteboard and draw lines to link them, or if you are using a regular wall, use masking tape to join one note to the next. In the best of all possible worlds, the walls of your office are painted with whiteboard paint

Figure 5.10

With quick and dirty process modeling, the business analyst uses Post-it notes to build an informal model of the work. Large notes are stuck on the wall and linked by masking tape. Naturally, interested stakeholders are partners in this modeling activity.



such as IdeaPaint and you can draw and stick and post notes wherever and of whatever you want.

The advantage of using Post-it notes to model the business process is that stakeholders are usually willing to become involved in moving the brightly colored notes around to model their process. If an activity is badly named, it can quickly be replaced by another note with a better name. If something is out of place, it can be moved around with little effort.

But perhaps the most important benefit of using this technique is that when the stakeholders start moving the Post-it notes around, they often discover ways of simplifying their business process. It is not uncommon to find business stakeholders saying things like “If we just move this activity over here, change that one, we could eliminate those two down there.” These ideas must be kept and brought into play when we look at the essence of the system and derive a new way of working.

The Post-its models range from simple models where the activities proceed more or less in a straight line, to the rather elaborate Post-it model shown in Figure 5.11.

Please keep in mind that while we are advocating the use of physical models, we are not advocating designing screens and delving into low-level detail—those actions are not appropriate at this stage. The intention of using quick and dirty models here is merely to build a more pliable model of the current and future business processes. Little is to be gained—and, in fact, much is to be lost—if business users turn their hand to designing screens at this time.

However, if you can avoid designing, you can make good use of sketches of screens and other interfaces.



**Figure 5.11**

A five-meter-long Post-it model built by the air traffic controllers during a workshop. This model is perhaps more elaborate than that needed by many businesses, but it helped the controllers to understand their process much better. *Photo: Neil Maiden.*

## Prototypes and Sketches

Prototyping and sketching—they are really the same thing—can be effective techniques for eliciting requirements. The basic idea is to sketch some proposed product, and then reverse engineer the requirements from the sketch. This is a particularly useful approach in the following situations:

- The product has not existed before, and it is difficult to visualize.
- The stakeholders for the product have no experience with either the kind of product or the proposed technology.
- The stakeholders have been doing their work for some time and are stuck in the way that they do it.
- The stakeholders are having trouble articulating their requirements.
- The requirements analysts are having trouble understanding what is required.
- The feasibility of a requirement is in doubt.

When gathering requirements, you ask your stakeholders to imagine what they need their future product to do. The results you uncover are limited

*The prototype makes the product real enough to prompt stakeholders to bring up requirements that might otherwise be missed.*

“Prototypes are requirements bait.”  
—Steve McMenamin

by the stakeholders’ imaginations and experiences, and by their ability to describe something that for the moment does not exist.

In contrast, a prototype gives the stakeholder something real, or at least something that has the appearance of reality. The prototype makes the product real enough for stakeholders to bring up requirements that might otherwise be missed. Our colleague Steve McMenamin refers to prototypes as “requirements bait”: When stakeholders see the functionality displayed by a prototype, it inspires them to bring up other requirements. In this way, by demonstrating possibilities through prototypes and giving stakeholders a seemingly real experience, you capture the additional requirements—sometimes there are quite a lot of them—that might otherwise have waited until the product was in use to come to the surface. (See Figure 5.12.)

Prototypes are also used to play out the consequences of requirements. You will inevitably come across a strange requirement that has a single advocate who swears he would be lost without it. Who knows whether his requirement is a great idea that will make the product better or merely a complex way of doing something that doesn’t need to be done? The prototype does. Building a prototype of hard-to-fathom requirements makes them visible; it gives everyone the opportunity to understand them, discuss them, possibly simplify them, and then decide whether their merits warrant their inclusion in the final product.

We have to emphasize that the prototypes and sketches discussed here are throwaway prototypes; they are not intended to evolve into the finished product. Of course, they might—but that possibility is incidental to your task of requirements gathering.

**Figure 5.12**

Requirements prototypes are used to display the functionality of a potential product. The prototype is intended to prompt stakeholders to tell you whether you have understood the needed functionality and, as a result of “using” the prototype, to give you additional requirements suggested by it.





In this discussion, we are using the terms “prototype” and “sketch” to mean some kind of simulation of a product that the stakeholders could use to do their work—the work that you envisage they might do in the future. Put simply, you show your prototype and possibly several alternative prototypes—to your stakeholders and ask if they could do their job using a product something like your simulation. If the answer is yes, then you capture the requirements demonstrated by that version of the prototype. If the answer is no, then you change the prototype and ask again.

The stakeholders who must decide whether your prototype is a reasonable demonstration of the proposed work face a somewhat difficult situation. Inevitably, they bring to the prototyping exercise some work-related baggage. They have been doing their job for some time (you probably wouldn’t be asking them to test your prototypes if they hadn’t), and their perception of their work may differ dramatically from yours. This difference in perspectives requires tact on your part to discover which aspects of reality are uppermost in the minds of the people you are dealing with. Which metaphors do they use for their work, and how do they envision themselves while they are doing their work? In addition, you are asking them how they feel about doing different work—the work that you are proposing for the future. Adopting the new product could mean jettisoning artifacts and ways of working that they feel quite comfortable with. It is no easy task for the stakeholders to turn away from their experience and their comfort, and to accept a new, as-yet-untried way of doing their work.

This means that you should always try to use prototyping techniques that conform, in some way, to the artifacts and experiences that are most familiar to the stakeholders. This means adjusting your prototyping approach for each work situation.

Let’s look at some of the possibilities available for requirements prototypes. We can then talk about how they might be gainfully employed. We’ll start with the simplest option: the low-fidelity prototype.

## Low-Fidelity Prototypes

By using familiar media, low-fidelity prototypes (we also call these “sketches”) help stakeholders concentrate on the subject matter. Such things as pencil and paper, whiteboards, flip charts, Post-it notes, index cards, and cardboard boxes can be employed to build effective low-fidelity prototypes (Figure 5.13). In fact, these prototypes may take advantage of anything that is part of the stakeholders’ everyday life and do not require an additional investment.

The low-fidelity prototype is a sketch, which is not meant to look all that much like the finished product—this is why such prototypes are called “low-fidelity.” This lack of faithfulness to the finished product is both good and bad. The good part is that no one will confuse your prototype with an actual

---

*Which aspects of reality are uppermost in the minds of the people you are dealing with? Which metaphors do they use for their work, and how do they envision themselves while they are doing their work?*

---



---

*Build a prototype that relates to the physical anchors in the user’s world.*

---

“To look at structure, the first prototypes are always paper.”

—Hugh Beyer and Karen Holtzblatt, *Contextual Design*



**Figure 5.13**

Effective prototyping tools do not have to be complex or expensive.



product—it is obviously built with a minimal investment in time, and most importantly it gives no indication that it is anything other than an easy-to-change mockup. The low-fidelity sketch encourages iteration, and it more or less indicates that you are not expecting to get the right answer on your first attempt. The bad part is that such a prototype sometimes requires more effort on the part of the stakeholder who is testing it to imagine that the whiteboard sketch might potentially be his new product. Nevertheless, the ability to make changes easily and the speed with which you can sketch out a prototype usually help you to bring things to life, thereby assisting stakeholders in taking more advantage of their imaginations.

---

*Prototyping is more convenient, and ultimately more accurate, if done one PUC at a time.*

---

We find that prototyping is more convenient, and ultimately more accurate, if the prototype involves a single business use case (BUC) or a single product use case (PUC); given that the prototype involves some simulated product, we will assume you are prototyping a PUC. We introduced business and product use cases in Chapter 4. Recall that a business use case is an amount of work, triggered by an external business event occurring or by a predetermined time being reached, that takes place in a single, continuous time frame. It also has a known, measurable, testable outcome. The part of the BUC that is to be done by the product is the PUC. Because of its single, continuous time frame, it provides you with an appropriate amount of work as the subject of your prototype.

Let's look at prototyping by examining an example use case. The "Monitor Untreated Roads" product use case, as shown in Figure 5.14, is suitable for our purposes. This use case is triggered periodically when it is time to monitor the road treatment: The engineer checks whether all the roads that have been scheduled to be treated with de-icing material have, in fact, been covered by one of the trucks. Your task is to find all the requirements for this part of the product.

Your aim in building a low-fidelity prototype of this BUC is to unearth the existing requirements that must be carried forward into the new product, along with the new, as yet undreamed-of requirements. Let's assume the engineers are currently working with a system that supplies them with a list of roads, and use that as a starting point.

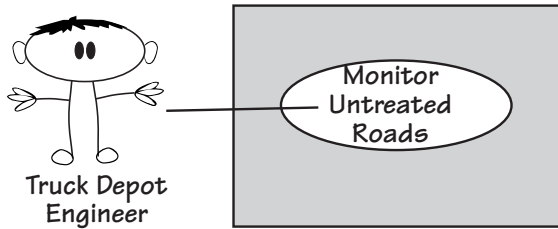


Figure 5.14

The truck depot engineers are responsible for ensuring that all roads in danger of freezing are treated with de-icing material. Use case number 11, “Monitor Untreated Roads,” represents this part of the work.

You can quickly sketch a low-fidelity prototype of the current situation, and then begin to explore improvements to it. The prototype focuses on what the product might do; for the moment, it ignores any implementation details.

When you are sketching a low-fidelity prototype, be quick. Demonstrate each requirements suggestion with another sketch, or several alternative sketches. You might intentionally serve up several prototypes of the same PUC, asking for the stakeholders’ preferences and possibly more suggestions for alternatives. Once your stakeholders see you are simulating potential ways of solving their problem and realize their input is not only welcome but also necessary, they will almost certainly help you by suggesting their own enhancements and requirements. Our experience is that once you get people involved by making the problem visible, they tend to become more creative and imaginative—so much so that sometimes your problem becomes one of keeping pace with their suggestions.

Get started by sketching what the stakeholders<sup>2</sup> might be doing when using the product. Ask the stakeholders what the product could do to contribute to their work, and note their ideas. In the “Monitor Untreated Roads” PUC, stakeholders would probably want the product to provide something like the following information:

- Roads that have been scheduled for treatment
- Scheduled roads that have been treated
- Relative positions of roads

Now ask the stakeholders what they would do with this information. At this point you are not trying to design a solution to this immediate problem, but rather want to explore ideas for what a potential product *might* do and what the work *might* be with that product:

- “What is the best way of presenting the needed information?”
- “Is the requested information the right information for the work being done?”
- “Could the product do more or less of the work?”

“Paper is eminently practical and meets the primary need: It makes it possible to express the structure of the system and makes it hard to overfocus on user interface detail.”

—Hugh Beyer and Karen Holtzblatt, *Contextual Design*

#### FOOTNOTE 2

We say “stakeholders” rather than “users,” because there are probably people other than users who are interested stakeholders for any use case.

When you are prototyping, assume that there are no technological limitations. Also, keep in mind that you are designing the work, not the product—you are capturing the things the product might do to help the stakeholders with their work.

Sketch pictures to elicit ideas from the stakeholders. If they are having trouble getting started, then inject some ideas of your own. Ask them to imagine that they are doing the job of monitoring untreated roads. Which other pieces of information would they need to do the job? Can your prototype provide it, or would they have to look elsewhere for information? Is all of the information in the current version of the prototype needed to do the job of monitoring untreated roads?

As the prototyping proceeds and the engineers see your sketches, you may hear the following kind of remark:

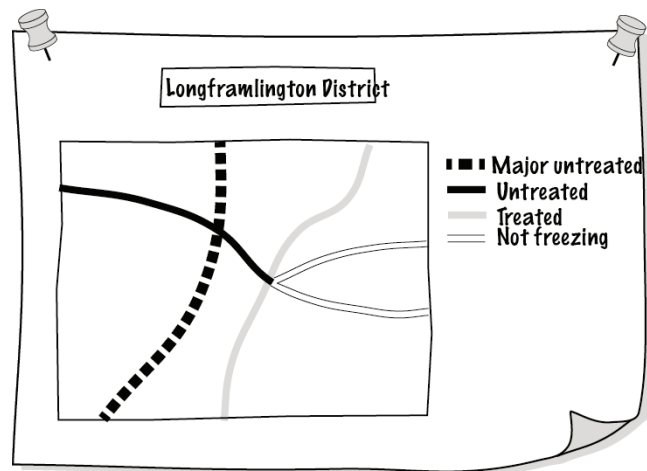
“That’s great. Now wouldn’t it be great if we could see the major roads that have not been treated for three hours? But only the major roads—the secondary ones don’t matter so much. Our current system can’t distinguish between major and secondary roads.”

How long would it take you to add that requirement to your sketch? About 10 seconds. How long would it have taken to modify the installed product if the engineers had asked for this requirement after delivery? Enough said.

By working with the engineers, you may generate a prototype that is quite different from the product they have at the moment. But that’s the point—to change the work, and to discover new and better ways of doing the work. By encouraging the engineers to tell you more and more about the job, you uncover requirements that otherwise would not come to light. (See Figure 5.15.)

**Figure 5.15**

By drawing this low-fidelity prototype on a flip chart or whiteboard, you help identify requirements for the truck depot engineers. They want the product to highlight the major roads within a district that have not been treated and are in a dangerous condition.



When you sketch a low-fidelity prototype, you demonstrate your ideas to the stakeholders and encourage them to iterate its development by changing and improving your ideas. Recall that a part of the requirements process is inventing a better way to do the work. The prototype is an invention vehicle with which you and your stakeholders to experiment to see how the proposed product could contribute to the new work.

---

*The low-fidelity prototype is not a work of art, but rather an idea-generating device.*

---

A low-fidelity prototype is informal and fast. No one will mistake it for the finished product. Because it is easy to change, stakeholders are willing to iterate, experiment, and investigate alternative ideas.

## High-Fidelity Prototypes

High-fidelity prototypes are built using software tools and have the appearance of working software products. They appear to do whatever the use case does, and they display most of the characteristics of working software products. Development of such a prototype gives stakeholders the opportunity to “use” a realistic-looking product and decide whether the product displays the correct requirements.

Because the prototype behaves as the stakeholders would expect the final product to behave, it gives them the chance to explore the *potential* of the product, ideally leading them to suggest improvements and new requirements. But herein lies a problem: Because the prototype looks so much like a working product, the stakeholders may be tempted to concentrate on its appearance and possibly forego making functional improvements. To do so, however, would defeat the purpose of requirements prototyping.

Putting aside the slight risk of stakeholders misunderstanding their role in the prototyping activity, the high-fidelity prototype has a lot to offer. It is interactive, which encourages users to explore its capabilities. The icons and data displayed on the screen are representative of the data and icons used to do the actual work. The stakeholders should feel at home with the prototype: They should be able to open windows, update data, and be told whether they have made an error. In other words, the prototype can be a realistic simulation of the real work.

A high-fidelity prototype allows you to create lifelike work situations, and ask the stakeholders to operate the prototype as if they were doing real work. By observing the stakeholders’ reactions and noting their comments, you find even more of their requirements.

An extensive set of high-fidelity prototyping tools is available. Interface builder types of tools incorporate functionality that allows you to easily build screens and simulate functionality. Many of these tools are backed by a database, thereby enabling you to construct small to medium-size working applications.

Content management systems such as Drupal, Alfresco, Joomla, and many more (a lot of them are available as open-source products) allow prototyping

of Web-based products. You should also check the CMS capabilities of SharePoint, as this is already available in many organizations. These tools are more than just prototyping tools—the final result may be the full-strength version of the product. Care must be taken to ensure the requirements for the other aspects of the product—security, maintainability, and so on—keep pace with the front end as it is being prototyped.

We will not attempt to list all products that could be used for high-fidelity prototypes. Any such list would be out of date before you picked up this book. We suggest searching the Web and, in particular, the open-source community (sourceforge.net) as the best way to find software suitable for requirements prototyping.

The high-fidelity prototype is effective for discovering usability requirements. As the user attempts to work with it, you observe which parts of the prototype are pleasing and easy to use, and which usability features need improvement. From these observations, the you can write most of the usability requirements.

But let's not get ahead of ourselves. In Chapter 8, *Starting the Solution*, we look at the interface of the potential product; this is where you concentrate on the usability aspects and other user-experience attributes of the product. High-fidelity prototypes are more commonly used at that point in the development process, so we will defer any more discussion of them until that chapter.

## Mind Maps



*We use mind maps all the time. We use them for taking notes during interviews, planning projects or actions, summarizing workshops—in fact, anytime we need to have a concise and intelligible record. Mind mapping is a skill that will benefit all business analysts.*

A mind map is a combination of drawing and text that attempts to represent information in the same way as your brain does. The mind map imitates your brain storage mechanism by using links between the words and pictures that represent the information.

Figure 5.16 shows a mind map—a friend drew this when she was planning her business. The business provides advice and recipes for people who dine alone. This market segment is unusual—lone diners usually do not have well-stocked larders and fridges, and sometimes they find it hard to buy food in quantities suitable for one person. Additionally, they might not be good cooks. Our friend's business, *Good Food for One*, provides recipes and advice on cooking when one is on one's own—you can see the business in the mind map.



Figure 5.16

A mind map drawn while planning the business Good Food for One. This Web-based business provides recipes, shopping advice, and cooking tips for people who cook and eat alone. Mind map by Joanna Kenneally. Used by permission.

Note the quick sketches and the hastily scribbled words in Figure 5.16. Joanna Kenneally, the mind map's author, was in a hurry. She wanted to start her business right away and needed a quick overview of what it would do; she wanted to record all her ideas, and see if they came together to form a viable business.

Mind maps are useful devices for organizing your thoughts. You can see the result of your thinking in one diagram—you get an overview and details at the same time—with each of your subtopics teased out to show divergence and connections. The mind map provides enough information—shown as keywords and links between those keywords—for you to get the overall picture. You can also decide to follow one of the branches, or make decisions based on having all the information laid out in a convenient format. Drawing small pictures on your mind map helps; they obviate the need to have as many words and have the advantage of being more easily remembered than words.

If you do not draw mind maps already, give them a try; nothing bad can happen to you. Start with the page in landscape format, and place your central subject in the center of it. The central idea should be some words, or a strong image, that tells you, or anyone who reads your mind map, what it is about. The first breakdown, or set of radiating lines, shows the major

---

*If you do not draw mind maps already, give them a try; nothing bad can happen.*

---



**READING**

Buzan, Tony, and Chris Griffiths. *Mind Maps for Business: Revolutionise Your Business Thinking and Practise*. BBC Active, 2009.

concepts, themes, ideas, or whatever subdivisions of your subject you choose. Write these ideas on the lines, using one or two words. Look for strong words that name and describe your ideas. And print rather than using cursive writing; the result will be both more readable and more memorable.

Use lines to make links between the ideas. We remember colors, so use colored lines for areas of the map that contain one theme. These lines can have arrowheads to denote direction, but most of the time the link between ideas is bidirectional.

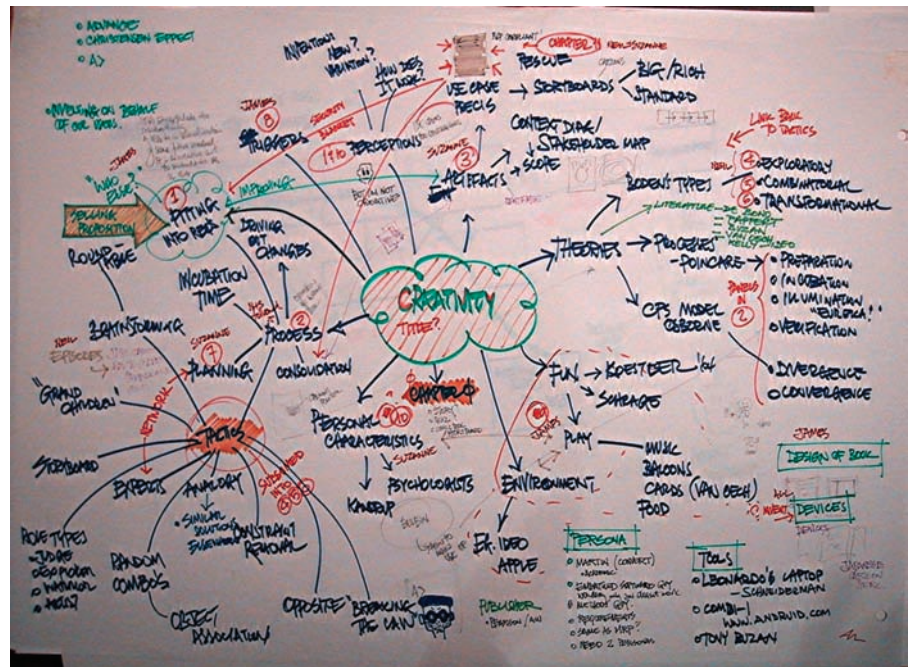
Mind maps cannot always be built from the center outward. Sometimes you get ideas, or hear things when you are taking notes, that have no connection to anything already in your map. Add it to the map anyway, because you will at some stage find a connection. Your mind map may not end up as organized and pretty as some examples you see, but it still makes sense to your mind—and that is the point. Figure 5.17 provides an example.

One last thing on the technique of drawing mind maps: Use the biggest sheet of paper or whiteboard you can find. Borrow a large drawing pad, or flip chart, or anything you can get your hands on. Spread out. Give your mind room to move. Amuse yourself with your mind map. And be as imaginative as you can.

You can, of course, buy software for drawing mind maps if you want to have them on your computer. There are plenty of mind mapping apps available. Your authors almost never use any of them, however, because it takes

**Figure 5.17**

A mind map your authors drew when planning an article on innovation and creativity.



away the sheer spontaneity of using paper and colored pencils. We photograph ours for posterity (as in the case of Figure 5.17).

An alternative technological approach is to use Wacom's Inkling as a way of recording your mind maps, along with your other sketches and handwritten notes. This technology uses plain paper on which you then draw or write, and faithfully records an electronic version of your drawing. It's an impressive tool for those of us who like to sketch and doodle.

We use mind maps for note taking when interviewing stakeholders about their requirements. The benefit of using mind maps in these situations is realized when your stakeholder tells you about features and functions of their work and the new product. Some of these are linked, and many are dependent on others. By drawing your interviews in mind map format, you become more likely to see the connections as well as to spot connections that the client has not made, but should be explained. Think of the mind map as a more versatile note-taking tool—versatile, because you can replace all of the text it takes to establish connections by simply drawing a line.

The best way of learning to draw mind maps is to start drawing mind maps. Use the tips described in this book, or gather examples from the Web to help. But mainly, just start drawing. You will soon see your mind map come to life.

## The Murder Book

Your authors spend too much time on airplanes. Reading light literature, particularly crime novels, is a favorite way to pass some of the long hours. Several authors, and in particular Michael Connelly and his Los Angeles detective Harry Bosch, make frequent reference to the *murder book*—a case file for a murder investigation. The murder book includes a trail of every document, interview, photograph, and other items of evidence collected during the investigation.

We often use this same approach with our business analysis assignments: We collect every document and other piece of “evidence” into a binder, sometimes several binders (see Figure 5.18). Like the detectives, our intention is to create a trail of items that can be referred to later from a central repository. Just as detectives often solve murders by going back through the



**Figure 5.18**

The murder book is a collection of documents and models generated by the requirements discovery project.



murder book, so the business analyst can often find wonderfully useful material from the project murder book.

The technique is simple: Add every document, interview note, model, mind map, user story, and paper prototype—in fact, everything—to the murder book. This assumes, of course, that you have paper artifacts. If everything is electronic, then you have an electronic murder book. Add your artifacts in chronological order, and that’s about all there is to the technique—except the part where you need to know something, and the murder book is where you look.

The murder book provides a trail of the story of the development. In the future when problems arise (and they will arise), the solutions to the new problems usually depend on knowing what took place and why things were decided previously in the project. The murder book is intended to be documentation of your development project.

We like this kind of documentation, particularly because it is free. There is no need to write extra material; just collect everything that has been written.

Sometimes you need to go a little further with this kind of record keeping and provide some kind of document register to make it easier for others to find your documents (or whatever form your gathered information is in).

With the availability of excellent search engines, we can see future potential for the majority of documentation being in this form. Instead of you having to spend a huge amount of time filing and cataloguing documents, the search engine finds what you want when you want it.

## Video and Photographs



*A video or photograph is a way of capturing some moments in time so that you can study them later. It is a particularly useful technique if you want to show the current work to people who cannot visit the stakeholders’ workplace. We also take photos of whiteboards—to show the progression of ideas—and find that we often refer back to these images and include some of them in documents.*

*Video and photographs are especially useful tools for distributed teams.*

“Video records help fill in the spaces between the paragraphs of more formal written documentation by retaining off-hand remarks, body language, facial expressions (like

Video can be used for studying the business. When users and business analysts participate in workshops and brainstorming sessions, for example, the proceedings may be videoed. Interviews and on-site observations may also be videoed. With this technique, video is used initially to record, and then confirm, the proceedings. In addition, you can show the video to developers who do not get the opportunity to meet face to face with the users.

Video can serve as an adjunct to interviewing and observing the users in their own workplace. Users have their own ways of accomplishing tasks. They have their own ways to categorize the information that they use, and

their own ways to solve problems that they have found worked well for them in their particular situation. Thus, by using video to capture the users at work, you also capture their ways of doing their jobs, their concerns, their preferences, and their idiosyncrasies.

Of course, video can be used in a more structured way as well. For example, you might select a business use case and ask the users to work through typical situations they encounter while you make a video recording of them. As they work, the users describe the special circumstances, the additional information they use, the exceptions, and so on. The shrugs, grimaces, and gestures that are normally lost when taking notes are faithfully recorded for later playback and dissection.

Obviously, you must ask permission before you begin to video someone. Keep in mind that whoever you are videoing will initially be very self-conscious in front of the camera, but subjects usually relax after a few minutes and accept the presence of the camera. Do not ask any important questions in the first five minutes, as the answers may be given for the benefit of the camera, and not for the benefit of accuracy.

We suggest that you try video in the following circumstances:

- Video users and developers participating in use case workshops and brainstorming.
- Video interviews and on-site observations.
- Video users at work.
- Video a business event. Ask the users to work through their typical response to an event, and to describe how they do their jobs.
- Send video to other members of a distributed team. We find the impact of video to be more involving for those who cannot see events firsthand.

If video does not seem practical in your environment, then use photographs. We always carry a small camera (the one in your phone is the most convenient) as part of our business analysis kit. During interviews, meetings, and workshops, we photograph stakeholders, whiteboards, flip charts, offices, manufacturing plants, and anything else that we think will be useful in helping us to discover the requirements. We often include photographs in minutes and progress reports. Photos are an effective way of giving people a detailed review without having to write a huge report.

You might also consider other recording devices such as the voice recorder on your smartphone, or a Livescribe smartpen, which acts as a pen and a sound recorder simultaneously.

*raised eyebrows), indications of stress in using the program, the ease of using the keyboard, and so on. Since it retains more information than language, which is simply an abstraction of the information on the tape, it retains the information more faithfully.*

—DeGrace and Stahl

#### READING

DeGrace, Peter, and Leslie Hulet Stahl. *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms*. Yourdon Press, 1990. This book has been around for a long time, but it has some relevant ideas on video to pass along.



## Wikis, Blogs, Discussion Forums



*This technique can apply to many types of projects. However, because it requires participation from a number of stakeholders, it is most effective for larger projects.*

People love to contribute. The success of Facebook, Twitter, Wikipedia, and blogs demonstrate that, given a chance and a forum, people are happy to spend a little time (sometimes a lot of time) recording their opinions and knowledge. These contributions take many forms—you have probably used several—but for the sake of brevity we shall refer to them all as being posted to *wikis*.

The basic idea of a wiki in requirements discovery is that anyone—and the point is largely that *anyone* can do it—can make a post or edit or add to whatever has already been posted. Some forums keep discussions threaded so you can see it unfold; some allow contributors to overwrite or reorganize whatever they find. Additionally, anyone can add hypertext links to other useful sources of information or make any other kind of change.

Wikis rely on technology, but it is technology readily available to all. You can buy or download free hosting solutions to host your requirements wiki. If your organization will not give you the server space, then many publicly available sites are available. If you take the latter route, check carefully before you post what could be sensitive commercial information in the public domain.

To start a wiki, you seed it with an outline of the proposed product and invite stakeholders to add their piece. Once someone posts an opinion on what the product should do, you will no doubt find others chipping in to support or refute the original posting. Others invariably have their say, which usually results in a substantial collection of information and opinions. Anyone can contribute to a wiki; if a contributor is not one of the stakeholders whom you identified in the blastoff, it doesn't matter. If a person has something to say, you want to hear it, and you want each of your contributors to see what the others are saying.

And it is free, or can be.

The Web is a bountiful source for requirements. Search for your domain of interest. You will likely uncover a lot of information on what other people have done in your domain and, if you are lucky, you will find information that you can readily convert to requirements for your product. At the very least, you will find papers and articles that provide valuable information about your domain. And, like wikis, searching the Web is free.

---

*The Web is a bountiful source for requirements. Search for your domain of interest. You will likely uncover a lot of information on what other people have done in your domain and, if you are lucky, you will find information that you can readily convert to requirements for your product.*

---

## Document Archeology

*Document archeology is a technique of searching through existing reports and files for underlying requirements. It is best used when you have some existing or legacy system, and plan on modifying or renewing it.*



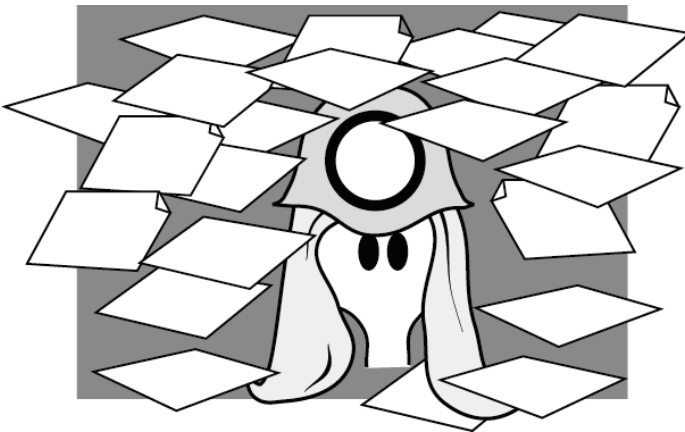
Document archeology involves determining the underlying requirements by inspecting the documents and files used by the business. It is not a complete technique, but rather should be used in conjunction with other techniques, and only with caution. Document archeology entails reverse engineering new requirements from the documents used or produced by the current work. Along the way, you look for requirements that should become part of the new product.

Obviously, not all of the old work will be carried forward; simply because something exists in a document does not mean that it automatically has to be part of a new system. Nevertheless, where a current system exists, it will always provide plenty of material that is grist for your requirements mill (see Figure 5.19).

Inspect the documents you have collected (for simplicity's sake, the term "document" means anything you have collected), looking for nouns, or "things." These can be column headings, named boxes on forms, or simply the name of a piece of data on the document.

For each "thing," ask several of these questions:

- What is the purpose of this thing?
- Who uses it and why?
- What are all the uses the system makes of this thing?
- Which business events use or reference this thing?



**Figure 5.19**

In document archeology, you start by collecting samples of all documents, reports, forms, files, and so on. Gather anything that is used to record or send information, including regular telephone calls. User manuals are rich sources of material—they describe a way to do work.

- Can this thing have an alphanumeric value? For example, is it a number, a code, or a quantity?
- If so, to which collection of things does it belong? (Data modeling enthusiasts will immediately recognize the need to find the entity or class that owns the attribute.)
- Does the document contain a repeating group of things?
- If so, what is the collection of things called?
- Can you find a link between things?
- Which process makes the connection between them?
- Which rules are attached to each thing? In other words, which piece of business policy covers the thing?
- Which processes ensure that these rules are obeyed?
- Which documents give users the most problems?

These questions will not, in themselves, reveal all the requirements for the product. They will, however, give you plenty of background material and suggest directions for further investigation.

When doing document archeology, you search for capabilities from the current work that are needed for the new product. This does not mean you have *carte blanche* to replicate the old system—you are, in fact, gathering requirements because you plan to build a new product. Nevertheless, an existing system will usually have some capabilities in common with its replacement.

But be warned: Just because a document is output from a current computer or manual system, it does not mean that it is correct, nor does it mean that it is what is wanted by your client. Perhaps the document serves no useful purpose, or it needs heavy modification before it can be reused successfully.

We suggest that you incorporate document archeology into your data modeling approach, because most of the answers from the questions listed earlier are commonly used in the latter discipline. Of course, some document archeology is used as a foundation for object-oriented development. The current documents, if used cautiously, may reveal the classes of the data. They may also reveal the attributes of data stored by the system, and sometimes suggest operations that should be performed on the data.

As a rule, we always keep artifacts—documents, printouts, lists, manuals, screens, and anything else that is printed or displayed—from our interviews because we often refer back to them (see the earlier section on the murder book). Make a habit of asking for a copy of any document or screen that is mentioned. It is also prudent to log the document with an ID and version number, as it ensures that everyone has the same idea as to the source of their knowledge.

## Family Therapy

Family therapists do not set out to make people *agree*. Instead, they aim to make it possible for people to hear and get an understanding of other individuals' positions, even if they do not agree with them. In other words, you should not expect every stakeholder to agree with every other stakeholder, but you should help the stakeholders as a group to accept that other people may disagree without necessarily being wrong, and that the need for choices and compromises will always arise. Early in the project, identify which mechanisms you will use to deal with these situations when they inevitably occur.

The field of family therapy is a rich source of ideas about how to work effectively with a diverse group of people—such as stakeholders in a requirements trawling process. We use ideas from family therapy as a way of helping us to listen to stakeholders and of providing a feedback loop to help avoid misinterpretations.

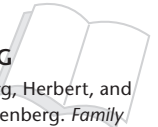
---

*Family therapists do not set out to make people agree. Instead, they aim to make it possible for people to hear and get an understanding of other individuals' positions.*

---

### READING

Goldenberg, Herbert, and Irene Goldenberg. *Family Therapy: An Overview*, eighth edition. Brooks Cole, 2012.



## Choosing the Best Trawling Technique

What is the best trawling technique? This is actually a trick question—there is no “best.” The technique you should use in any given situation depends on several factors. The first, and most important, is that you feel comfortable with the technique. There is nothing quite as dismaying to a stakeholder audience as to have their business analyst stumbling while trying to learn a technique and at the same time use it. If you are a right-brain person and prefer graphical approaches, then that is what you should do. If you are more of a left-brain thinker—that is, your preference is for serial thinking—then text-based techniques such as scenarios would be more attractive to you.

Similarly, your stakeholders must feel comfortable with whatever technique you have selected. If a stakeholder cannot understand the model you are showing him, then it is simply the wrong model to use.

There are a couple of other considerations:

- **Geography:** When you are selecting a technique, keep in mind the locations of the stakeholders. Some techniques are more readily adaptable to dispersed teams, or when you have to deal remotely with your stakeholders.
- **Legacy:** How much of the current implementation has to remain there? How much does this constraint affect possible future implementations? If you have to carry forward a considerable legacy, then some of the more abstract techniques will not be suitable.
- **Abstraction:** Are you in a situation where you can deal with the business in an abstract or essential way? In some cases your stakeholders are not abstract thinkers, and you have to concentrate more on techniques to deal with physical realities. In contrast, if you are dealing

with the desired future state of the business, then techniques involving abstraction are favored.

- **Knowledge:** What is the domain of your work area? It is worth taking this factor into account when you are selecting a trawling technique. Some domains such as engineering are very physical, while others such as finance are more conceptual and you can use more abstract techniques.

Table 5.1 lists various trawling techniques and summarizes their strengths. Some of these techniques have not yet been discussed; we will encounter them in later chapters. We suggest this table as a useful place for getting started in selecting a technique, or for finding alternative techniques to the ones you are using at the moment.

**Table 5.1**

**Trawling Techniques and Their Strengths**

<b>Trawling Technique</b>	<b>Strengths</b>
<i>Business events</i>	<i>Partition the work for external demands</i>
<i>Current situation modeling</i>	<i>Examines the legacy system for reusable requirements</i>
<i>Apprenticing</i>	<i>Spends time working with an expert</i>
<i>Structures and patterns</i>	<i>Identify reusable requirements</i>
<i>Interviewing</i>	<i>Can focus on detailed issues</i>
<i>Essence</i>	<i>Finds the real problem</i>
<i>Business use case workshops</i>	<i>Focus the relevant stakeholders on the best response to the business event</i>
<i>Creativity workshops</i>	<i>Team discovers innovative requirements</i>
<i>Brainstorming</i>	<i>Facilitates creativity and invention</i>
<i>Personas</i>	<i>Use a composite virtual character to represent the user/customer</i>
<i>Mind mapping</i>	<i>An effective planning/note-taking technique</i>
<i>Wikis</i>	<i>Online forums through which stakeholders can contribute</i>
<i>Scenarios</i>	<i>Show the functionality of a use case</i>
<i>Low-fidelity prototypes</i>	<i>Discover undreamed-of requirements</i>
<i>High-fidelity prototypes</i>	<i>Discover usability requirements</i>
<i>Document archeology</i>	<i>Uses evidence from existing documents and files</i>
<i>Family therapy</i>	<i>Uses techniques from psychology to help stakeholders to understand a variety of viewpoints and to make choices clear.</i>

We also refer you to the owls in the margins; they indicate when particular techniques are most effective.

Always use techniques with which you and your stakeholders are comfortable. The best results come when you and the people you are dealing with feel at ease with the way you are working together to discover requirements. If a technique is not working for you, then try a different one.

## Finally . . .

Trawling, as we have discussed it here, is concerned with uncovering and understanding the business. At first, the business analyst's study is mainly concerned with the current work. This study should be done as rapidly as possible, and should be no more than is needed to reach a consensus among the stakeholders as to what the work actually is. The techniques presented in this chapter are tools intended to help with this study. Moreover, you have to do this work yourself—no tool has yet been invented that can do it for you.

We have not spent much time with the most critical tools for a business analyst—the ones attached to either side of your head, and the big gray one between those. *Listening* is the most important technique in requirements gathering. If you can listen to what your owner and your other stakeholders are saying, and if you can *think* about what they say and understand what they mean, then the tools described in this chapter will be useful. Conversely, if you don't listen and don't think, then you are highly unlikely to uncover the product that the user really wants.

The trawling techniques are communication tools; they help you open a dialog with your stakeholders and provide the feedback that is so essential to good communication. Use them well and wisely.

“ You have two ears and one mouth. I suggest that you use them in that proportion.”  
—G. K. Chesterton