CHAPTER 3

REQUIREMENTS DETERMINATION

One of the first activities of an analyst is to determine the business requirements for a new system. This chapter begins by presenting the requirements definition, a document that lists the new system's capabilities. It then describes how to analyze requirements using requirements analysis strategies and how to gather requirements using interviews, JAD sessions, questionnaires, document analysis, and observation. The chapter also describes a set of alternative requirements-documentation techniques and describes the system proposal document that pulls everything together.

OBJECTIVES

- Understand how to create a requirements definition
- Become familiar with requirements-analysis techniques
- Understand when to use each requirements-analysis technique
- Understand how to gather requirements using interviews, JAD sessions, questionnaires, document analysis, and observation
- Understand the use of concept maps, story cards, and task lists as requirements-documentation techniques
- Understand when to use each requirements-gathering technique
- Be able to begin creating a system proposal

INTRODUCTION

The systems development process aids an organization in moving from the current system (often called the *as-is system*) to the new system (often called the *to-be system*). The output of planning, discussed in Chapter 2, is the system request, which provides general ideas for the to-be system, defines the project's scope, and provides the initial workplan. Analysis takes the general ideas in the system request and refines them into a detailed requirements definition (this chapter), functional models (Chapter 4), structural models (Chapter 5), and behavioral models (Chapter 6) that together form the *system proposal*. The system proposal also includes revised project management deliverables, such as the feasibility analysis and the workplan (Chapter 2).

The output of analysis, the system proposal, is presented to the approval committee, who decides if the project is to continue. If approved, the system proposal moves into design, and its elements (requirements definition and functional, structural, and behavioral models) are used as inputs to the steps in design. This further refines them and defines in much more detail how the system will be built.

The line between analysis and design is very blurry. This is because the deliverables created during analysis are really the first step in the design of the new system. Many of the major design decisions for the new system are found in the analysis deliverables. It is

important to remember that the deliverables from analysis are really the first step in the design of the new system.

In many ways, because it is here that the major elements of the system first emerge, the requirements-determination step is the single most critical step of the entire system development process. During requirements determination, the system is easy to change because little work has been done yet. As the system moves through the system development process, it becomes harder and harder to return to requirements determination and to make major changes because of all of the rework that is involved. Several studies have shown that more than half of all system failures are due to problems with the requirements.[1] This is why the iterative approaches of object-oriented methodologies are so effective—small batches of requirements can be identified and implemented in incremental stages, allowing the overall system to evolve over time.

REQUIREMENTS DETERMINATION

The purpose of *requirements determination* is to turn the very high-level explanation of the business requirements stated in the system request into a more precise list of requirements that can be used as inputs to the rest of analysis (creating functional, structural, and behavioral models). This expansion of the requirements ultimately leads to the design of the system.

Defining a Requirement

A *requirement* is simply a statement of what the system must do or what characteristic it must have. During analysis, requirements are written from the perspective of the businessperson, and they focus on the "what" of the system. Because they focus on the needs of the business user, they are usually called *business requirements* (and sometimes user requirements). Later in design, business requirements evolve to become more technical, and they describe how the system will be implemented. Requirements in design are written from the developer's perspective, and they are usually called *system requirements.*

We want to stress that there is no black-and-white line dividing a business requirement and a system requirement—and some companies use the terms interchangeably. The important thing to remember is that a requirement is a statement of what the system must do, and requirements will change over time as the project moves from inception to elaboration to construction. Requirements evolve from detailed statements of the business capabilities that a system should have to detailed statements of the technical way the capabilities will be implemented in the new system.

Requirements can be either functional or nonfunctional in nature. A *functional requirement* relates directly to a process a system has to perform or information it needs to contain. For example, requirements stating that a system must have the ability to search for available inventory or to report actual and budgeted expenses are functional requirements. Functional requirements flow directly into the creation of functional, structural, and behavioral models that represent the functionality of the evolving system (see Chapters 4, 5, and 6).

Nonfunctional requirements refer to behavioral properties that the system must have, such as performance and usability. The ability to access the system using a Web browser is considered a nonfunctional requirement. Nonfunctional requirements can influence the rest of analysis (functional, structural, and behavioral models) but often do so only indirectly; nonfunctional requirements are used primarily in design when decisions are made about the database, the user interface, the hardware and software, and the system's underlying physical architecture.

[1] For example, see *The Scope of Software Development Project Failures* (Dennis, MA: The Standish Group, 1995).

Nonfunctional requirements describe a variety of characteristics regarding the system: operational, performance, security, and cultural and political. Operational requirements address issues related to the physical and technical requirements in which the system will operate. Performance requirements address issues related to the speed, capacity, and reliability of the system. Security requirements deal with issues with regard to who has access to the system and under what specific circumstances. Cultural and political requirements deal with issues related to the cultural, political factors and legal requirements that affect the system. These characteristics do not describe business processes or information, but they are very important in understanding what the final system should be like. Nonfunctional requirements primarily affect decisions that will be made during the design of a system. We will return to this topic later in the book when we discuss design (see Chapters 9, 10, and 11).

One area of information systems development that focused on differentiating functional and nonfunctional requirements is *software quality*. There have been many different models proposed to measure the quality of software. However, virtually all of them differentiate functional and nonfunctional requirements. From a quality perspective, *functional* quality is related to the degree that the software meets the functional requirements, i.e., how much of the actual problem is solved by the software solution provided. Whereas, the nonfunctional requirements are associated with the efficiency, maintainability, portability, reliability, reusability, testability, and usability quality dimensions. As stated above, the nonfunctional related dimensions are associated primarily with the actual detailed design and implementation of the system.

When considering ISO 9000 compliance, quality dimensions are further decomposed into those that the user can see (*external*) and those that the user cannot see (*internal*). The external nonfunctional dimensions include efficiency, reliability, and usability, whereas the internal nonfunctional dimensions include maintainability, portability, reusability, and testability. From a user perspective, the external dimensions are more important. If the system is simply too difficult to use, regardless how well the system solves the problem, the user will simply not use the system. In other words, from a user's perspective, for an information system to be successful, the system must not only meet the functional specification, but it must also meet the external nonfunctional specifications. From a developer perspective, the internal dimensions are also important. For example, given that successful systems tend to be long-lived and multiplatform, both the maintainability and portability dimensions can have strategic implications for the system being developed. Also, given the *agile* development approaches being used in industry today, the development of reusable and testable software is crucial.

Three additional topics that have influenced information system requirements are the Sarbanes-Oxley Act, COBIT (Control OBjectives for Information and related Technology) compliance and Capability Maturity Model compliance. Depending on the system being considered, these three topics could affect the definition of a system's functional requirements, nonfunctional requirements, or both. The Sarbanes-Oxley Act, for example, mandates additional functional and nonfunctional requirements. These include additional security concerns (nonfunctional) and specific information requirements that management must now provide (functional). When developing financial information systems, information system developers should be sure to include Sarbanes-Oxley expertise in the development team. Moreover, a client could insist on COBIT compliance or that a specific Capability Maturity Model level had been reached in order for the firm to be considered as a possible vendor to supply the system under consideration. Obviously, these types of requirements add to the nonfunctional requirements. Further discussion of these topics is beyond the scope of this book.[2]

[2] A concise discussion of the Sarbanes-Oxley Act is presented in G. P. Lander, *What is Sarbanes-Oxley?* (New York: McGraw-Hill, 2004). A good reference for Sarbanes-Oxley Act-based security requirements is D. C. Brewer, *Security Controls for Sarbanes-Oxley Section 404 IT Compliance: Authorization, Authentication, and Access* (Indianapolis, IN: Wiley, 2006). For detailed information on COBIT, see www.isaca.org; for ISO 9000, see www.iso.org; and for details on the Capability Maturity Model, see www.sei.cmu.edu/cmmi/.

Another recent topic that influences requirements for some systems is globalization. For example, a global information supply chain generates a large number of additional nonfunctional requirements. If the necessary operational environments do not exist for a mobile solution to be developed, it is important to adapt the solution to the local environment. Or, it may not be reasonable to expect to deploy a high-technology-based solution in an area that does not have the necessary power and communications infrastructure. In some cases, we may need to consider supporting some parts of the global information supply chain with manual—rather than automated—systems.

Manual systems have an entirely different set of requirements that create different performance expectations and additional security concerns. Furthermore, cultural and political concerns are potentially paramount. A simple example that affects the design of user interfaces is the proper use of color on forms (on a screen or paper). Different cultures interpret different colors differently. In other words, in a global, multicultural business environment, addressing cultural concerns goes well beyond simply having a multilingual user interface. We must be able to adapt the global solution to the local realities. Friedman refers to these concerns as glocalization.[3] Otherwise, we will simply create another example of a failed information system development project.

Requirements Definition

The requirements definition report—usually just called the *requirements definition*—is a straightforward text report that simply lists the functional and nonfunctional requirements in an outline format. Figure 3-1 shows a sample requirements definition for an appointment system for a typical doctor's office. Notice it contains both functional and nonfunctional requirements. The functional requirements include managing appointments, producing schedules, and recording the availability of the individual doctors. The nonfunctional requirements include items such as the expected amount of time that it takes to store a new appointment, the need to support wireless printing, and which types of employees have access to the different parts of the system.

The requirements are numbered in a legal or outline format so that each requirement is clearly identified. The requirements are first grouped into functional and nonfunctional requirements; within each of those headings, they are further grouped by the type of nonfunctional requirement or by function.

Sometimes business requirements are prioritized on the requirements definition. They can be ranked as having high, medium, or low importance in the new system, or they can be labeled with the version of the system that will address the requirement (e.g., release 1, release 2, release 3). This practice is particularly important when using object-oriented methodologies since they deliver systems in an incremental manner.

The most obvious purpose of the requirements definition is to provide the information needed by the other deliverables in analysis, which include functional, structural, and behavioral models, and to support activities in design. The most important purpose of the requirements definition, however, is to define the scope of the system. The document describes to the analysts exactly what the system needs to end up doing. When discrepancies arise, the document serves as the place to go for clarification.

Determining Requirements

Determining requirements for the requirements definition is both a business task and an information technology task. In the early days of computing, there was a presumption that

[3] T. L. Friedman, *The World is Flat: A Brief History of the Twenty-First Century, Updated and Expanded Edition.* (New York: Farrar, Straus, and Giroux, 2006.)

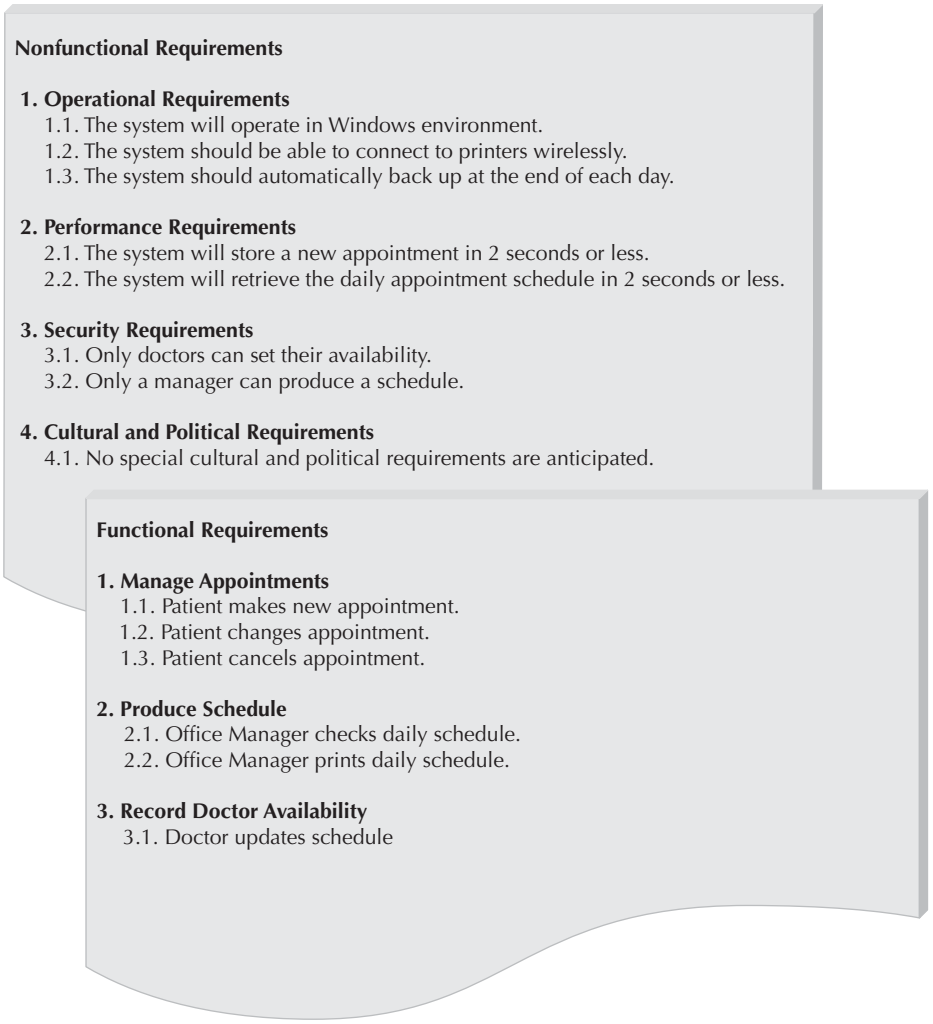


FIGURE 3-1
Sample Requirements Definition

the systems analysts, as experts with computer systems, were in the best position to define how a computer system should operate. Many systems failed because they did not adequately address the true business needs of the users. Gradually, the presumption changed so that the users, as the business experts, were seen as being the best position to define how a computer system should operate. However, many systems failed to deliver performance benefits because users simply automated an existing inefficient system, and they failed to incorporate new opportunities offered by technology.

Therefore, the most effective approach is to have both business people and analysts working together to determine business requirements. Sometimes, however, users don't know exactly what they want, and analysts need to help them discover their needs. A set of strategies has become popular to help analysts do problem analysis, root cause analysis, duration analysis, activity-based costing, informal benchmarking, outcome analysis, technology analysis, and activity elimination. Analysts can use these tools when they need to guide the users in explaining what is wanted from a system. These strategies work similarly. They help users critically examine the current state of systems and processes (the as-is system), identify exactly what needs to change, and develop a concept for a new system (the to-be system).

Although these strategies enable the analyst to help users create a vision for the new system, they are not sufficient for extracting information about the detailed business requirements that are needed to build it. Therefore, analysts use a portfolio of requirements-gathering techniques to acquire information from users. The analyst has many techniques from which to choose: interviews, questionnaires, observation, joint application development (JAD), and document analysis. The information gathered using these techniques is critically analyzed and used to craft the requirements definition report.

Creating a Requirements Definition

Creating a requirements definition is an iterative and ongoing process whereby the analyst collects information with requirements-gathering techniques (e.g., interviews, document analysis), critically analyzes the information to identify appropriate business requirements for the system, and adds the requirements to the requirements definition report. The requirements definition is kept up to date so that the project team and business users can refer to it and get a clear understanding of the new system.

To create a requirements definition, the project team first determines the kinds of functional and nonfunctional requirements that they will collect about the system (of course, these may change over time). These become the main sections of the document. Next, the analysts use a variety of requirements-gathering techniques to collect information, and they list the business requirements that were identified from that information. Finally, the analysts work with the entire project team and the business users to verify, change, and complete the list and to help prioritize the importance of the requirements that were identified.

This process continues throughout analysis, and the requirements definition evolves over time as new requirements are identified and as the project moves into later phases of the Unified Process. Beware: The evolution of the requirements definition must be carefully managed. The project team cannot keep adding to the requirements definition, or the system will keep growing and growing and never get finished. Instead, the project team carefully identifies requirements and evaluates which ones fit within the scope of the system. When a requirement reflects a real business need but is not within the scope of the current system or current release, it is either added on a list of future requirements or given a low priority. The management of requirements (and system scope) is one of the hardest parts of managing a project.

Real-World Problems with Requirements Determination

Avison and Fitzgerald provide us with a set of problems that can arise with regard to determining the set of requirements with which to be dealt.[4] First, the analyst might not have access to the correct set of users to uncover the complete set of requirements. This can lead to requirements being missed, misrepresented, and/or overspecified. Second, the specification of the requirements may be inadequate. This can be especially true with the lightweight techniques associated with agile methodologies. Third, some requirements are simply unknowable at the beginning of a development process. However, as the system is developed, the users and analysts will get a better understanding of both the domain issues and the applicable technology. This can cause new functional and nonfunctional requirements to be identified and current requirements to evolve or be canceled. Iterative and incremental-based development methodologies, such as the Unified Process and agile, can help in this case. Fourth, verifying and validating of requirements can be very difficult. We take up this topic in the chapters that deal with the creation of functional (Chapter 4), structural (Chapter 5), and behavioral (Chapter 6) models.

[4] See D. Avison and G. Fitzgerald, *Information Systems Development: Methodologies, Techniques, & Tools,* 4th Ed. (London: McGraw-Hill, 2006).

REQUIREMENTS ANALYSIS STRATEGIES

Before the project team can determine what requirements are appropriate for a given system, there needs to be a clear vision of the kind of system that will be created and the level of change that it will bring to the organization. The basic process of *analysis* is divided into three steps: understanding the as-is system, identifying improvements, and developing requirements for the to-be system.

Sometimes the first step (i.e., understanding the as-is system) is skipped or is performed in a cursory manner. This happens when no current system exists, if the existing system and processes are irrelevant to the future system, or if the project team is using a RAD or agile development methodology in which the as-is system is not emphasized. Newer RAD, agile, and object-oriented methodologies, such as phased development, prototyping, throwaway prototyping, extreme programming, and Scrum (see Chapter 1) focus almost exclusively on improvements and the to-be system requirements, and they spend little time investigating the current as-is system.

Requirements analysis strategies help the analyst lead users through the analysis steps so that the vision of the system can be developed. Requirements analysis strategies and requirements-gathering techniques go hand in hand. Analysts use requirements-gathering techniques to collect information; requirements analysis strategies drive the kind of information that is gathered and how it is ultimately analyzed. The requirements analysis strategies and requirements gathering happen concurrently and are complementary activities.

To move the users from the as-is system to the to-be system, an analyst needs strong *critical thinking skills*. Critical thinking is the ability to recognize strengths and weaknesses and recast an idea in an improved form, and critical thinking skills are needed to really understand issues and develop new business processes. These skills are also needed to thoroughly examine the results of requirements gathering, to identify business requirements, and to translate those requirements into a concept for the new system.

Problem Analysis

The most straightforward (and probably the most commonly used) requirements-analysis technique is *problem analysis*. Problem analysis means asking the users and managers to identify problems with the as-is system and to describe how to solve them in the to-be system. Most users have a very good idea of the changes they would like to see, and most are quite vocal about suggesting them. Most changes tend to solve problems rather than capitalize on opportunities, but the latter is possible as well. Improvements from problem analysis tend to be small and incremental (e.g., provide more space in which to type the customer's address; provide a new report that currently does not exist).

This type of improvement often is very effective at improving a system's efficiency or ease of use. However, it often provides only minor improvements in business value—the new system is better than the old, but it may be hard to identify significant monetary benefits from the new system.

Root Cause Analysis

The ideas produced by problem analysis tend to be solutions to problems. All solutions make assumptions about the nature of the problem, assumptions that might or might not be valid. In our experience, users (and most people in general) tend to quickly jump to solutions without fully considering the nature of the problem. Sometimes the solutions are appropriate, but many times they address a *symptom* of the problem, not the true problem or *root cause* itself.[5]

[5] Two good books that discuss the difficulty in finding the root causes to problems are: E. M. Goldratt and J. Cox, *The Goal* (Croton-on-Hudson, NY: North River Press, 1986); E. M. Goldratt, *The Haystack Syndrome* (Croton-on-Hudson, NY: North River Press, 1990).

For example, suppose a firm notices that its users report inventory stock-outs. The cost of inventory stock-outs can be quite significant. In this case, since they happen frequently, customers could find another source for the items that they are purchasing from the firm. It is in the firm's interest to determine the underlying cause and not simply provide a knee-jerk reaction such as arbitrarily increasing the amount of inventory kept on hand. In the business world, the challenge lies in identifying the root cause—few real-world problems are simple. The users typically propose a set of causes for the problem under consideration. The solutions that users propose can address either symptoms or root causes, but without a careful analysis, it is difficult to tell which one is addressed.

Root cause analysis, therefore, focuses on problems, not solutions. The analyst starts by having the users generate a list of problems with the current system and then prioritize the problems in order of importance. Starting with the most important, the users and/or the analysts then generate all the possible root causes for the problems. Each possible root cause is investigated (starting with the most likely or easiest to check) until the true root causes are identified. If any possible root causes are identified for several problems, those should be investigated first, because there is a good chance they are the real root causes influencing the symptom problems. In our example, there are several possible root causes:

- The firm's supplier might not be delivering orders to the firm in a timely manner.
- There could be a problem with the firm's inventory controls.
- The reorder level and quantities could be set wrong.

Sometimes, using a hierarchical chart to represent the causal relationships helps with the analysis. As Figure 3-2 shows, there are many possible root causes that underlie the higher-level causes identified. The key point in root cause analysis is always to challenge the obvious.

Duration Analysis

Duration analysis requires a detailed examination of the amount of time it takes to perform each process in the current as-is system. The analysts begin by determining the total amount of time it takes, on average, to perform a set of business processes for a typical input. They then time each of the individual steps (or subprocesses) in the business process. The time to

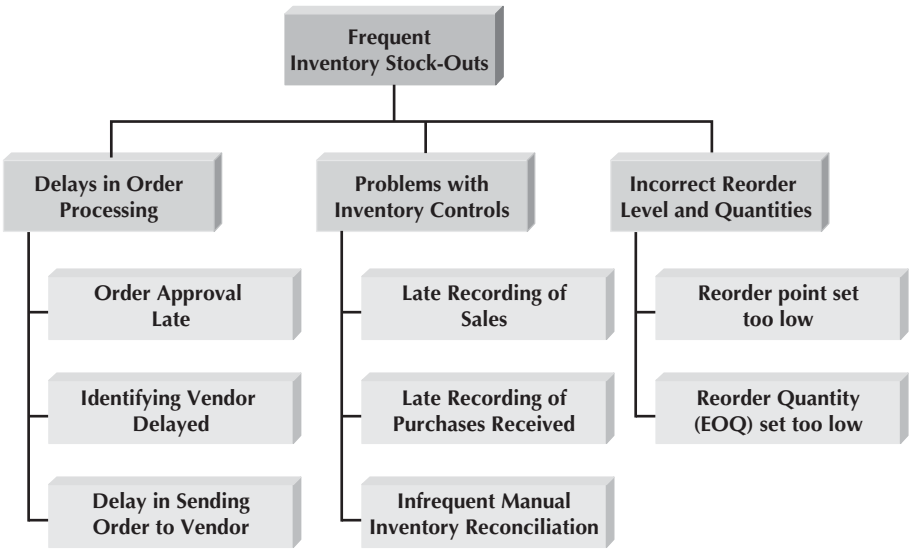


FIGURE 3-2 Root Cause Analysis for Inventory Stock-Outs

complete the basic step is then totaled and compared to the total for the overall process. A significant difference between the two—and in our experience the total time often can be 10 or even 100 times longer than the sum of the parts—indicates that this part of the process is badly in need of a major overhaul.

For example, suppose that the analysts are working on a home mortgage system and discover that on average, it takes thirty days for the bank to approve a mortgage. They then look at each of the basic steps in the process (e.g., data entry, credit check, title search, appraisal) and find that the total amount of time actually spent on each mortgage is about eight hours. This is a strong indication that the overall process is badly broken, because it takes thirty days to perform one day's work.

These problems probably occur because the process is badly fragmented. Many different people must perform different activities before the process finishes. In the mortgage example, the application probably sits on many people's desks for long periods of time before it is processed.

Processes in which many different people work on small parts of the inputs are prime candidates for *process integration* or *parallelization*. Process integration means changing the fundamental process so that fewer people work on the input, which often requires changing the processes and retraining staff to perform a wider range of duties. Process parallelization means changing the process so that all the individual steps are performed at the same time. For example, in the mortgage application case, there is probably no reason that the credit check cannot be performed at the same time as the appraisal and title check.

Activity-Based Costing

Activity-based costing is a similar analysis; it examines the cost of each major process or step in a business process rather than the time taken.[6] The analysts identify the costs associated with each of the basic functional steps or processes, identify the most costly processes, and focus their improvement efforts on them.

Assigning costs is conceptually simple. Analysts simply examine the direct cost of labor and materials for each input. Materials costs are easily assigned in a manufacturing process, whereas labor costs are usually calculated based on the amount of time spent on the input and the hourly cost of the staff. However, as you may recall from a managerial accounting course, there are indirect costs, such as rent, depreciation, and so on, that also can be included in activity costs.

Informal Benchmarking

Benchmarking refers to studying how other organizations perform a business process in order to learn how your organization can do something better. Benchmarking helps the organization by introducing ideas that employees may never have considered but that have the potential to add value.

Informal benchmarking is fairly common for customer-facing business processes (i.e., processes that interact with the customer). With informal benchmarking, the managers and analysts think about other organizations or visit them as customers to watch how the business process is performed. In many cases, the business studied may be a known leader in the industry or simply a related firm.

[6] Many books have been written on activity-based costing. Useful ones include K. B. Burk and D. W. Webster, *Activity-Based Costing* (Fairfax, VA: American Management Systems, 1994); D. T. Hicks, *Activity-Based Costing: Making It Work for Small and Mid-sized Companies* (New York: Wiley, 1998). The two books by Eli Goldratt mentioned previously (*The Goal* and *The Haystack Syndrome*) also offer unique insights into costing.

Outcome Analysis

Outcome analysis focuses on understanding the fundamental outcomes that provide value to customers. Although these outcomes sound as though they should be obvious, they often are not. For example, consider an insurance company. One of its customers has just had a car accident. What is the fundamental outcome from the *customer's* perspective? Traditionally, insurance companies have answered this question by assuming the customer wants to receive the insurance payment quickly. To the customer, however, the payment is only a *means* to the real outcome: a repaired car. The insurance company might benefit by extending its view of the business process past its traditional boundaries to include not paying for repairs but performing the repairs or contracting with an authorized body shop to do them.

With this approach, system analysts encourage the managers and project sponsor to pretend they are customers and to think carefully about what the organization's products and services enable the customers to do—and what they *could* enable the customer to do.

Technology Analysis

Many major changes in business since the turn of the century have been enabled by new technologies. *Technology analysis* starts by having the analysts and managers develop a list of important and interesting technologies. Then the group systematically identifies how every technology could be applied to the business process and identifies how the business would benefit. It is important to note the technology analysis in no way implies adopting technology for technology's sake. Rather the focus is on using new technologies to meet the goals of the organization.

Activity Elimination

Activity elimination is exactly what it sounds like. The analysts and managers work together to identify how the organization could eliminate each activity in the business process, how the function could operate without it, and what effects are likely to occur. Initially, managers are reluctant to conclude that processes can be eliminated, but this is a force-fit exercise in that they must eliminate each activity. In some cases, the results are silly; nonetheless, participants must address every activity in the business process.

REQUIREMENTS-GATHERING TECHNIQUES

An analyst is very much like a detective (and business users are sometimes like elusive suspects). He or she knows that there is a problem to be solved and therefore must look for clues that uncover the solution. Unfortunately, the clues are not always obvious (and are often missed), so the analyst needs to notice details, talk with witnesses, and follow leads just as Sherlock Holmes would have done. The best analysts thoroughly gather requirements using a variety of techniques and make sure that the current business processes and the needs for the new system are well understood before moving into design. Analysts don't want to discover later that they have key requirements wrong—such surprises late in the development process can cause all kinds of problems.

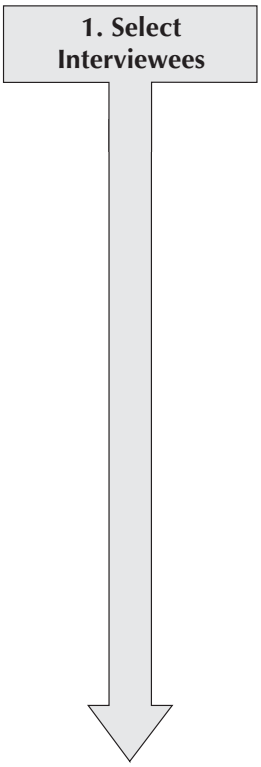
The requirements-gathering process is used for building political support for the project and establishing trust and rapport between the project team building the system and the users who ultimately will choose to use or not use the system. Involving someone in the process implies that the project teams view that person as an important resource and value his or her opinions. All the key stakeholders (the people who can affect the system or who will be affected by the system) must be included in the requirements-gathering process. The

stakeholders might include managers, employees, staff members, and even some customers and suppliers. If a key person is not involved, that individual might feel slighted, which can cause problems during implementation (e.g., How could they have developed the system without my input?).

The second challenge of requirements gathering is choosing the way(s) information is collected. There are many techniques for gathering requirements that vary from asking people questions to watching them work. In this section, we focus on the five most commonly used techniques: interviews, JAD sessions (a special type of group meeting), questionnaires, document analysis, and observation. Each technique has its own strengths and weaknesses, many of which are complementary, so most projects use a combination of techniques.[7]

Interviews

An interview is the most commonly used requirements-gathering technique. After all, it is natural—if you need to know something, you usually ask someone. In general, interviews are conducted one-on-one (one interviewer and one interviewee), but sometimes, owing to time constraints, several people are interviewed at the same time. There are five basic steps to the interview process: selecting interviewees, designing interview questions, preparing for the interview, conducting the interview, and postinterview follow-up.[8]



The first step in interviewing is to create an *interview schedule* listing who will be interviewed, when, and for what purpose (see Figure 3-3). The schedule can be an informal list that is used to help set up meeting times or a formal list that is incorporated into the workplan. The people who appear on the interview schedule are selected based on the analyst's information needs. The project sponsor, key business users, and other members of the project team can help the analyst determine who in the organization can best provide important information about requirements. These people are listed on the interview schedule in the order in which they should be interviewed.

People at different levels of the organization have varying perspectives on the system, so it is important to include both managers who manage the processes and staff who actually perform the processes to gain both high-level and low-level perspectives on an issue. Also, the kinds of interview subjects needed can change over time. For example, at the start of the project, the analyst has a limited understanding of the as-is business process. It is common to begin by interviewing one or two senior managers to get a strategic view and then to move to midlevel managers who can provide broad, overarching information about the business process and the expected role of the system being developed. Once the analyst has a good understanding of the big picture, lower-level managers and staff members can fill in the exact details of how the process works. Like most other things about systems analysis, this is an iterative process—starting with senior managers, moving to midlevel managers, then staff members, back to midlevel managers, and so on, depending upon what information is needed along the way.

It is quite common for the list of interviewees to grow, often by 50 to 75 percent. As people are interviewed, more information that is needed and additional people who can provide the information will probably be identified.

[7] Some excellent books that address the importance of gathering requirements and various techniques include Alan M. Davis, *Software Requirements: Objects, Functions, & States, Revision* (Englewood Cliffs, NJ: Prentice Hall, 1993); Gerald Kotonya and Ian Sommerville, *Requirements Engineering* (Chichester, England: Wiley, 1998); Dean Leffingwell and Don Widrig, *Managing Software Requirements: A Unified Approach* (Reading, MA: Addison-Wesley, 2000).

[8] A good book on interviewing is that by Brian James, *The Systems Analysis Interview* (Manchester, England: NCC Blackwell, 1989).

Name	Position	Purpose of Interview	Meeting
Andria McClellan	Director, Accounting	Strategic vision for new accounting system	Mon., March 1 8:00–10:00 AM
Jennifer Draper	Manager, Accounts Receivable	Current problems with accounts receivable process; future goals	Mon., March 1 2:00–3:15 PM
Mark Goodin	Manager, Accounts Payable	Current problems with accounts payable process; future goals	Mon., March 1 4:00–5:15 PM
Anne Asher	Supervisor, Data Entry	Accounts receivable and payable processes	Wed., March 3 10:00–11:00 AM
Fernando Merce	Data Entry Clerk	Accounts receivable and payable processes	Wed., March 3 1:00–3:00 PM

FIGURE 3-3
Sample Interview Schedule

2. Design Interview Questions

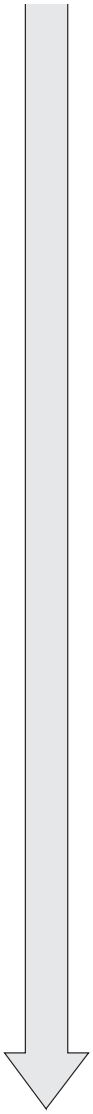
There are three types of interview questions: closed-ended questions, open-ended questions, and probing questions. *Closed-ended questions* are those that require a specific answer. They are similar to multiple-choice or arithmetic questions on an exam (see Figure 3-4). Closed-ended questions are used when an analyst is looking for specific, precise information (e.g., how many credit card requests are received per day). In general, precise questions are best. For example, rather than asking, Do you handle a lot of requests? it is better to ask, How many requests do you process per day? Closed-ended questions enable analysts to control the interview and obtain the information they need. However, these types of questions don't uncover *why* the answer is the way it is, nor do they uncover information that the interviewer does not think to ask for ahead of time.

Open-ended questions are those that leave room for elaboration on the part of the interviewee. They are similar in many ways to essay questions that you might find on an exam (see Figure 3-4 for examples). Open-ended questions are designed to gather rich information and give the interviewee more control over the information that is revealed during the interview. Sometimes the information that the interviewee chooses to discuss uncovers information that is just as important as the answer (e.g., if the interviewee talks only about other departments when asked for problems, it may suggest that he or she is reluctant to admit his or her own problems).

The third type of question is the *probing question*. Probing questions follow up on what has just been discussed in order to learn more, and they often are used when the interviewer is unclear about an interviewee's answer. They encourage the interviewee to expand on or to confirm information from a previous response, and they signal that the interviewer is listening and is interested in the topic under discussion. Many beginning analysts are reluctant to use probing questions because they are afraid that the interviewee might be offended at being challenged or because they believe it shows that they didn't understand what the interviewee said. When done politely, probing questions can be a powerful tool in requirements gathering.

In general, an interviewer should not ask questions about information that is readily available from other sources. For example, rather than asking what information is used to perform to a task, it is simpler to show the interviewee a form or report (see the section on document analysis) and ask what information on it is used. This helps focus the interviewee on the task and saves time, because the interviewee does not need to describe the information detail—he or she just needs to point it out on the form or report.

No type of question is better than another, and a combination of questions is usually used during an interview. At the initial stage of an IS development project, the as-is process can



be unclear, so the interview process begins with *unstructured interviews,* interviews that seek broad and roughly defined information. In this case, the interviewer has a general sense of the information needed but has few closed-ended questions to ask. These are the most challenging interviews to conduct because they require the interviewer to ask open-ended questions and probe for important information on the fly.

As the project progresses, the analyst comes to understand the business process much better and needs very specific information about how business processes are performed (e.g., exactly how a customer credit card is approved). At this time, the analyst conducts *structured interviews,* in which specific sets of questions are developed before the interviews. There usually are more closed-ended questions in a structured interview than in the unstructured approach.

No matter what kind of interview is being conducted, interview questions must be organized into a logical sequence so that the interview flows well. For example, when trying to gather information about the current business process, it can be useful to move in logical order through the process or from the most important issues to the least important.

There are two fundamental approaches to organizing the interview questions: top down or bottom up (see Figure 3-5). With the *top-down interview*, the interviewer starts with broad, general issues and gradually works toward more-specific ones. With the *bottom-up interview*, the interviewer starts with very specific questions and moves to broad questions. In practice, analysts mix the two approaches, starting with broad, general issues, moving to specific questions, and then returning to general issues.

The top-down approach is an appropriate strategy for most interviews (it is certainly the most common approach). The top-down approach enables the interviewee to become accustomed to the topic before he or she needs to provide specifics. It also enables the interviewer to understand the issues before moving to the details because the interviewer might not have sufficient information at the start of the interview to ask very specific questions. Perhaps most importantly, the top-down approach enables the interviewee to raise a set of big-picture issues before becoming enmeshed in details, so the interviewer is less likely to miss important issues.

One case in which the bottom-up strategy may be preferred is when the analyst already has gathered a lot of information about issues and just needs to fill in some holes with details. Bottom-up interviewing may be appropriate if lower-level staff members feel threatened or unable to answer high-level questions. For example, How can we improve customer service? might be too broad a question for a customer service clerk, whereas a specific question is readily answerable (e.g., How can we speed up customer returns?). In any event, all interviews should begin with noncontroversial questions and then gradually move into more contentious issues after the interviewer has developed some rapport with the interviewee.

Types of Questions	Examples
Closed-ended questions	• How many telephone orders are received per day? • How do customers place orders? • What information is missing from the monthly sales report?
Open-ended questions	• What do you think about the current system? • What are some of the problems you face on a daily basis? • What are some of the improvements you would like to see in a new system?
Probing questions	• Why? • Can you give me an example? • Can you explain that in a bit more detail?

FIGURE 3-4
Three Types of Questions

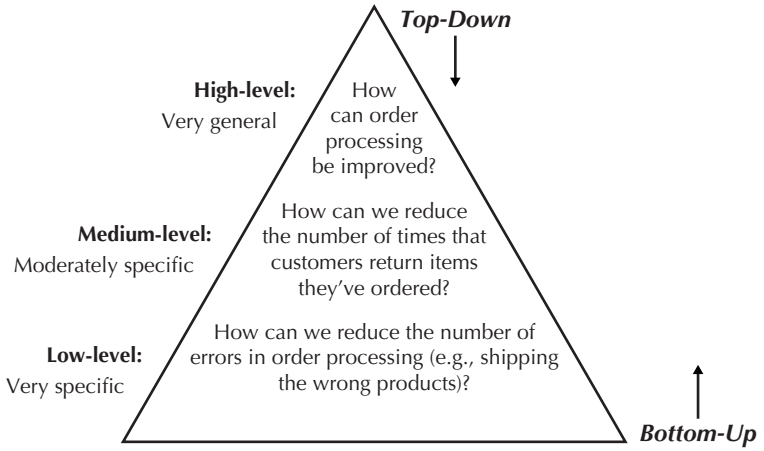


FIGURE 3-5 Top-Down and Bottom-Up Questioning Strategies

3. Prepare for the Interview

It is important to prepare for the interview in the same way that you would prepare to give a presentation. The interviewer should have a general interview plan listing the questions to be asked in the appropriate order, should anticipate possible answers and provide follow-up with them, and should identify segues between related topics. The interviewer should confirm the areas in which the interviewee has knowledge so as not to ask questions that the interviewee cannot answer. Review the topic areas, the questions, and the interview plan, and clearly decide which have the greatest priority in case time runs short.

In general, structured interviews with closed-ended questions take more time to prepare than unstructured interviews. Some beginning analysts prefer unstructured interviews, thinking that they can wing it. This is very dangerous and often counterproductive, because any information not gathered in the first interview will require follow-up efforts, and most users do not like to be interviewed repeatedly about the same issues.

The interviewer should be sure to prepare the interviewee as well. When the interview is scheduled, the interviewee should be told the reason for the interview and the areas that will be discussed far enough in advance so that he or she has time to think about the issues and organize his or her thoughts. This is particularly important when the interviewer is an outsider to the organization and for lower-level employees, who often are not asked for their opinions and who may be uncertain about why they are being interviewed.

4. Conduct the Interview

The first goal is to build rapport with the interviewee, so that he or she trusts the interviewer and is willing to tell the whole truth, not just give the answers that he or she thinks are wanted. The interviewer should appear to be a professional and unbiased, independent seeker of information. The interview should start with an explanation of why the interviewer is there and why he or she has chosen to interview the person; then the interviewer should move into the planned interview questions.

It is critical to carefully record all the information that the interviewee provides. In our experience, the best approach is to take careful notes—write down *everything* the interviewee says, even if it does not appear immediately relevant. The interviewer shouldn't be afraid to ask the person to slow down or to pause while writing, because this is a clear indication that the interviewee's information is important. One potentially controversial issue is whether or not to tape-record an interview. Recording ensures that the interviewer does not miss important

points, but it can be intimidating for the interviewee. Most organizations have policies or generally accepted practices about the recording of interviews, so they should be determined before an interview. If the interviewer is worried about missing information and cannot tape the interview, then he or she can bring along a second person to take detailed notes.

As the interview progresses, it is important to understand the issues that are discussed. If the interviewer does not understand something, he or she should ask for clarification. The interviewer should not be afraid to ask dumb questions, because the only thing worse than appearing dumb is to be dumb by not understanding something. If the interviewer doesn't understand something during the interview, he or she certainly won't understand it afterwards. Jargon should be recognized and defined; any jargon not understood should be clarified. One good strategy to increase understanding during an interview is to periodically summarize the key points that the interviewee is communicating. This avoids misunderstandings and also demonstrates that the interviewer is listening.

Finally, facts should be separated from opinion. The interviewee may say, for example, We process too many credit card requests. This is an opinion, and it is useful to follow this up with a probing question requesting support for the statement (e.g., Oh, how many do you process in a day?). It is helpful to check the facts because any differences between the facts and the interviewee's opinions can point out key areas for improvement. Suppose the interviewee complains about a high or increasing number of errors, but the logs show that errors have been decreasing. This suggests that errors are viewed as a very important problem that should be addressed by the new system, even if they are declining.

As the interview draws to a close, the interviewee should have time to ask questions or provide information that he or she thinks is important but was not part of the interview plan. In most cases, the interviewee has no additional concerns or information, but in some cases this leads to unanticipated, but important, information. Likewise, it can be useful to ask the interviewee if there are other people who should be interviewed. The interview should end on time (if necessary, some topics can be omitted or another interview can be scheduled).

As a last step in the interview, the interviewer should briefly explain what will happen. The interviewer shouldn't prematurely promise certain features in the new system or a specific delivery date, but he or she should reassure the interviewee that his or her time was well spent and very helpful to the project.

5. Post-Interview Follow-up

After the interview is over, the analyst needs to prepare an *interview report* that describes the information from the interview (Figure 3-6). The report contains *interview notes,* information that was collected over the course of the interview and is summarized in a useful format. In general, the interview report should be written within forty-eight hours of the interview, because the longer the interviewer waits, the more likely he or she is to forget information.

Often, the interview report is sent to the interviewee with a request to read it and inform the analyst of clarifications or updates. The interviewee needs to be convinced that the interviewer genuinely wants his or her corrections to the report. Usually there are few changes, but the need for any significant changes suggests that a second interview will be required. Never distribute someone's information without prior approval.

Joint Application Development (JAD)

JAD is an information-gathering technique that allows the project team, users, and management to work together to identify requirements for the system. IBM developed the JAD technique in the late 1970s, and it is often the most useful method for collecting information from users.[9]

[9] More information on JAD can be found in J. Wood and D. Silver, *Joint Application Development* (New York: Wiley, 1989); Alan Cline, "Joint Application Development for Requirements Collection and Management," http://www.carolla.com/wp-jad.htm.

Interview Notes Approved by: Linda Estey

Person Interviewed: Linda Estey,
Director, Human Resources

Interviewer: Barbara Wixom

Purpose of Interview:

- Understand reports produced for Human Resources by the current system
- Determine information requirements for future system

Summary of Interview:

- Sample reports of all current HR reports are attached to this report. The information that is not used and missing information are noted on the reports.
- Two biggest problems with the current system are:
  1. The data are too old (the HR Department needs information within two days of month end; currently, information is provided to them after a three-week delay)
  2. The data are of poor quality (often reports must be reconciled with departmental HR database)
- The most common data errors found in the current system include incorrect job level information and missing salary information.

Open Items:

- Get current employee roster report from Mary Skudrna (extension 4355).
- Verify calculations used to determine vacation time with Mary Skudrna.
- Schedule interview with Jim Wack (extension 2337) regarding the reasons for data quality problems.

Detailed Notes: See attached transcript.

FIGURE 3-6 Interview Report

Capers Jones claims that JAD can reduce scope creep by 50 percent and prevent the system's requirements from being too specific or too vague, both of which cause trouble during later stages of the development process.[10]

JAD is a structured process in which ten to twenty users meet together under the direction of a *facilitator* skilled in JAD techniques. The facilitator sets the meeting agenda and guides the discussion but does not join in the discussion as a participant. He or she does not provide ideas or opinions on the topics under discussion so as to remain neutral during the session. The facilitator must be an expert in both group-process techniques and systems-analysis and design techniques. One or two *scribes* assist the facilitator by recording notes, making copies, and so on. Often the scribes use computers and CASE tools to record information as the JAD session proceedings.

The JAD group meets for several hours, several days, or several weeks until all the issues have been discussed and the needed information is collected. Most JAD sessions take place in a specially prepared meeting room, away from the participants' offices so that they are not interrupted. The meeting room is usually arranged in a U-shape so that all participants can easily see each other. At the front of the room (the open part of the U), are a whiteboard, flip chart, and/or overhead projector for use by the facilitator leading the discussion.

[10] See Kevin Strehlo, "Catching up with the Jones and 'Requirement' Creep," *Infoworld* (July 29, 1996); Kevin Strehlo, "The Makings of a Happy Customer: Specifying Project X," *Infoworld* (November 11, 1996).

PRACTICAL TIP

3-1 Developing Interpersonal Skills

Interpersonal skills are skills that enable you to develop rapport with others, and they are very important for interviewing. They help you to communicate with others effectively. Some people develop good interpersonal skills at an early age; they simply seem to know how to communicate and interact with others. Other people are less lucky and need to work hard to develop their skills.

Interpersonal skills, like most skills, can be learned. Here are some tips:

- **Don't worry, be happy.** Happy people radiate confidence and project their feelings on others. Try interviewing someone while smiling and then interviewing someone else while frowning and see what happens.
- **Pay attention.** Pay attention to what the other person is saying (which is harder than you might think). See how many times you catch yourself with your mind on something other than the conversation at hand.
- **Summarize key points.** At the end of each major theme or idea that someone explains, repeat the key points back to the speaker (e.g., Let me make sure I understand. The key issues are. . . ."). This demonstrates that you consider the information important, and it also forces you to pay attention (you can't repeat what you didn't hear).
- **Be succinct.** When you speak, be succinct. The goal in interviewing (and in much of life) is to learn, not to impress. The more you speak, the less time you give to others.
- **Be honest.** Answer all questions truthfully, and if you don't know the answer, say so.
- **Watch body language (yours and theirs).** The way a person sits or stands conveys much information. In general, a person who is interested in what you are saying sits or leans forward, makes eye contact, and often touches his or her face. A person leaning away from you or with an arm over the back of a chair is uninterested. Crossed arms indicate defensiveness or uncertainty, and steepling (sitting with hands raised in front of the body with fingertips touching) indicates a feeling of superiority.

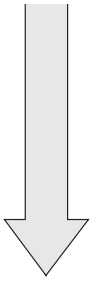
JAD suffers from the traditional problems associated with groups: Sometimes people are reluctant to challenge the opinions of others (particularly their boss), a few people often dominate the discussion, and not everyone participates. In a fifteen-member group, for example, if everyone participates equally, then each person can talk for only four minutes each hour and must listen for the remaining fifty-six minutes—not a very efficient way to collect information.

A new form of JAD called *electronic JAD*, or *e-JAD*, attempts to overcome these problems by using groupware. In an e-JAD meeting room, each participant uses special software on a networked computer to send anonymous ideas and opinions to everyone else. In this way, all participants can contribute at the same time without fear of reprisal from people with differing opinions. Initial research suggests that e-JAD can reduce the time required to run JAD sessions by 50 to 80 percent.[11] A good JAD approach follows a set of five steps.

1. Select Participants

JAD participants are selected in the same way as are interview participants, based on the information they can contribute in order to provide a broad mix of organizational levels and to build political support for the new system. The need for all JAD participants to be away from their office at the same time can be a major problem. The office might need to be closed or operate with a skeleton staff until the JAD sessions are complete.

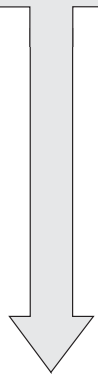
[11] For more information on e-JAD, see A. R. Dennis, G. S. Hayes, and R. M. Daniels, "Business Process Modeling with Groupware," *Journal of Management Information Systems* 15, no. 4 (1999): 115–142.



Ideally, the participants who are released from regular duties to attend the JAD sessions should be the very best people in that business unit. However, without strong management support, JAD sessions can fail because those selected to attend the JAD session are people who are less likely to be missed (i.e., the least competent people).

The facilitator should be someone who is an expert in JAD or e-JAD techniques and, ideally, someone who has experience with the business under discussion. In many cases, the JAD facilitator is a consultant external to the organization because the organization might not have a recurring need for JAD or e-JAD expertise. Developing and maintaining this expertise in-house can be expensive.

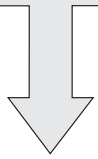
2. Design a JAD Session



JAD sessions can run from as little as half a day to several weeks, depending upon the size and scope of the project. In our experience, most JAD sessions tend to last five to ten days, spread over a three-week period. Most e-JAD sessions tend to last one to four days in a one-week period. JAD and e-JAD sessions usually go beyond collecting information and move into analysis. For example, the users and the analysts collectively can create analysis deliverables, such as the functional models or the requirements definition.

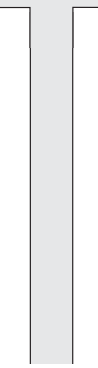
JAD sessions usually are designed and structured using the same principles as interviews. Most JAD sessions are designed to collect specific information from users, and this requires developing a set of questions before the meeting. One difference between JAD and interviewing is that all JAD sessions are structured—they *must* be carefully planned. In general, closed-ended questions are seldom used because they do not spark the open and frank discussion that is typical of JAD. In our experience, it is better to proceed top down in JAD sessions when gathering information. Typically thirty minutes is allocated to each separate agenda item, and frequent breaks are scheduled throughout the day because participants tire easily.

3. Preparing for a JAD Session



As with interviewing, it is important to prepare the analysts and participants for a JAD session. Because the sessions can go beyond the depth of a typical interview and are usually conducted off-site, participants may be more concerned about how to prepare. It is important that the participants understand what is expected of them. If the goal of the JAD session, for example, is to develop an understanding of the current system, then participants can bring procedure manuals and documents with them. If the goal is to identify improvements for a system, then before they come to the JAD session they can think about how they would improve the system.

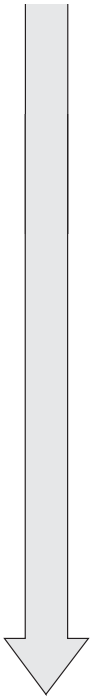
4. Conducting a JAD Session



Most JAD sessions follow a formal agenda, and most have formal *ground rules* that define appropriate behavior. Common ground rules include following the schedule, respecting others' opinions, accepting disagreement, and ensuring that only one person talks at a time.

The role of a JAD facilitator can be challenging. Many participants come to a JAD session with strong feelings about the system to be discussed. Channeling these feelings so that the session moves forward in a positive direction and getting participants to recognize and accept—but not necessarily agree on—opinions and situations different from their own requires significant expertise in systems analysis and design, JAD, and interpersonal skills. Few systems analysts attempt to facilitate JAD sessions without being trained in JAD techniques, and most apprentice with a skilled JAD facilitator before they attempt to lead their first session.

The JAD facilitator performs three key functions. First, he or she ensures that the group sticks to the agenda. The only reason to digress from the agenda is when it becomes clear to the facilitator, project leader, and project sponsor that the JAD session has produced some new information that is unexpected and requires the JAD session (and perhaps the project) to move in a new direction. When participants attempt to divert the discussion away from the



agenda, the facilitator must be firm but polite in leading discussion back to the agenda and getting the group back on track.

Second, the facilitator must help the group understand the technical terms and jargon that surround the system-development process and help the participants understand the specific analysis techniques used. Participants are experts in their area, or their part of the business, but they are not experts in systems analysis. The facilitator must, therefore, minimize the learning required and teach participants how to effectively provide the right information.

Third, the facilitator records the group's input on a public display area, which can be a whiteboard, flip chart, or computer display. He or she structures the information that the group provides and helps the group recognize key issues and important solutions. The facilitator must remain neutral at all times and simply help the group through the process. The moment the facilitator offers an opinion on an issue, the group will see him or her not as a neutral party but rather as someone who could be attempting to sway the group into some predetermined solution.

However, this does not mean that the facilitator should not try to help the group resolve issues. For example, if two items appear to be the same to the facilitator, the facilitator should not say, "I think these may be similar." Instead, the facilitator should ask, "Are these similar?" If the group decides they are, the facilitator can combine them and move on. However, if the group decides they are not similar (despite what the facilitator believes), the facilitator should accept the decision and move on. The group is *always* right, and the facilitator has no opinion.

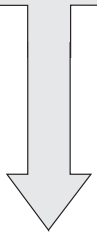
5. Post-JAD Follow-up

As with interviews, a JAD *post-session report* is prepared and circulated among session attendees. The post-session report is essentially the same as the interview report in Figure 3-6. Because the JAD sessions are longer and provide more information, it usually takes a week or two after the JAD session before the report is complete.

Questionnaires

A questionnaire is a set of written questions used to obtain information from individuals. Questionnaires are often used when there is a large number of people from whom information and opinions are needed. In our experience, questionnaires are a common technique with systems intended for use outside the organization (e.g., by customers or vendors) or for systems with business users spread across many geographic locations. Most people automatically think of paper when they think of questionnaires, but today more questionnaires are being distributed in electronic form, either via e-mail or on the Web. Electronic distribution can save a significant amount of money as compared to distributing paper questionnaires. A good process to use when using questionnaires follows four steps.

1. Select Participants



As with interviews and JAD sessions, the first step is to identify the individuals to whom the questionnaire will be sent. However, it is not usual to select every person who could provide useful information. The standard approach is to select a *sample,* or subset, of people who are representative of an entire group. Sampling guidelines are discussed in most statistics books, and most business schools include courses that cover the topic, so we do not discuss it here. The important point in selecting a sample, however, is to realize that not everyone who receives a questionnaire will actually complete it. On average, only 30 to 50 percent of paper and e-mail questionnaires are returned. Response rates for Web-based questionnaires tend to be significantly lower (often only 5 to 30 percent).

PRACTICAL TIP

Managing Problems in JAD Sessions

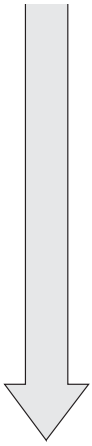
I have run more than a hundred JAD sessions and have learned several standard "facilitator tricks." Here are some common problems and some ways to deal with them.

- **Domination.** The facilitator should ensure that no one person dominates the group discussion. The only way to deal with someone who dominates is head on. During a break, approach the person, thank him or her for his or her insightful comments, and ask the person to help you make sure that others also participate.
- **Noncontributors.** Drawing out people who have participated very little is challenging because you want to bring them into the conversation so that they will contribute again. The best approach is to ask a direct factual question that you are certain they can answer. And it helps to ask the question in a long way to give them time to think. For example, "Pat, I know you've worked shipping orders a long time. You've probably been in the shipping department longer than anyone else. Could you help us understand exactly what happens when an order is received in shipping?"
- **Side discussions.** Sometimes participants engage in side conversations and fail to pay attention to the group. The easiest solution is simply to walk close to the people and continue to facilitate right in front of them. Few people will continue a side conversion when you are two feet from them and the entire group's attention is on you and them.
- **Agenda merry-go-round.** The merry-go-round occurs when a group member keeps returning to the same issue every few minutes and won't let go. One solution is to let the person have five minutes to ramble on about the issue while you carefully write down every point on a flip chart or computer file. This flip chart or file is then posted conspicuously on the wall. When the person brings up the issue again, you interrupt them, walk to the paper and ask them what to add. If they mention something already on the list, you quickly interrupt, point out that it is there, and ask what other information to add. Don't let them repeat the same point, but write any new information.
- **Violent agreement.** Some of the worst disagreements occur when participants really agree on the issues but don't realize that they agree because they are using different terms. An example is arguing whether a glass is half empty or half full; they agree on the facts but can't agree on the words. In this case, the facilitator has to translate the terms into different words and find common ground so the parties recognize that they really agree.
- **Unresolved conflict.** In some cases, participants don't agree and can't understand how to determine what alternatives are better. You can help by structuring the issue. Ask for criteria by which the group will identify a good alternative (e.g., "Suppose this idea really did improve customer service. How would I recognize the improved customer service?"). Then once you have a list of criteria, ask the group to assess the alternatives using them.
- **True conflict.** Sometimes, despite every attempt, participants just can't agree on an issue. The solution is to postpone the discussion and move on. Document the issue as an open issue and list it prominently on a flip chart. Have the group return to the issue hours later. Often the issue will have resolved itself by then and you haven't wasted time on it. If the issue cannot be resolved later, move it to the list of issues to be decided by the project sponsor or some other more senior member of management.
- **Humor.** Humor is one of the most powerful tools a facilitator has and thus must be used judiciously. The best JAD humor is always in context; never tell jokes but take the opportunity to find the humor in the situation.

Alan Dennis

2. Designing a Questionnaire

Because the information on a questionnaire cannot be immediately clarified for a confused respondent, developing good questions is critical for questionnaires. Questions on questionnaires must be very clearly written and leave little room for misunderstanding, so closed-ended questions tend to be most commonly used. Questions must clearly enable the analyst to separate facts from opinions. Opinion questions often ask respondents the extent to which they agree or disagree (e.g., Are network problems common?), whereas factual questions seek more

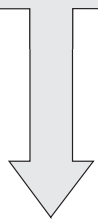


precise values (e.g., How often does a network problem occur: once an hour, once a day, once a week?). See Figure 3-7 for guidelines on questionnaire design.

Perhaps the most obvious issue—but one that is sometimes overlooked—is to have a clear understanding of how the information collected from the questionnaire will be analyzed and used. This issue must be addressed before the questionnaire is distributed, because it is too late afterward.

Questions should be relatively consistent in style, so that the respondent does not have to read instructions for each question before answering it. It is generally good practice to group related questions together to make them simpler to answer. Some experts suggest that questionnaires should start with questions important to respondents, so that the questionnaire immediately grabs their interest and induces them to answer it. Perhaps the most important step is to have several colleagues review the questionnaire and then pretest it with a few people drawn from the groups to whom it will be sent. It is surprising how often seemingly simple questions can be misunderstood.

3. Administering the Questionnaire



The key issue in administering the questionnaire is getting participants to complete the questionnaire and send it back. Dozens of marketing research books have been written about ways to improve response rates. Commonly used techniques include clearly explaining why the questionnaire is being conducted and why the respondent has been selected, stating a date by which the questionnaire is to be returned, offering an inducement to complete the questionnaire (e.g., a free pen), and offering to supply a summary of the questionnaire responses. Systems analysts have additional techniques to improve response rates inside the organization, such as personally handing out the questionnaire and personally contacting those who have not returned them after a week or two, as well as requesting the respondents' supervisors to administer the questionnaires in a group meeting.

4. Questionnaire Follow-up

It is helpful to process the returned questionnaires and develop a questionnaire report soon after the questionnaire deadline. This ensures that the analysis process proceeds in a timely fashion and that respondents who requested copies of the results receive them promptly.

Document Analysis

Project teams often use document analysis to understand the as-is system. Under ideal circumstances, the project team that developed the existing system will have produced documentation that was then updated by all subsequent projects. In this case, the project team can start by reviewing the documentation and examining the system itself.

Unfortunately, many systems are not well documented because project teams fail to document their projects along the way, and when the projects are over, there is no time to go back and document. Therefore, there might not be much technical documentation about the current systems available, or it might not contain updated information about recent system changes. However, many helpful documents do exist in an organization: paper reports,

- Begin with nonthreatening and interesting questions.
- Group items into logically coherent sections.
- Do not put important items at the very end of the questionnaire.
- Do not crowd a page with too many items.
- Avoid abbreviations.
- Avoid biased or suggestive items or terms.
- Number questions to avoid confusion.
- Pretest the questionnaire to identify confusing questions.
- Provide anonymity to respondents.

FIGURE 3-7
Good Questionnaire Design

memorandums, policy manuals, user-training manuals, organization charts, forms, and, of course, the user interface with the existing system.

But these documents tell only part of the story. They represent the *formal system* that the organization uses. Quite often, the real, or *informal, system* differs from the formal one, and these differences, particularly large ones, give strong indications of what needs to be changed. For example, forms or reports that are never used should probably be eliminated. Likewise, boxes or questions on forms that are never filled in (or are used for other purposes) should be rethought. See Figure 3-8 for an example of how a document can be interpreted.

The most powerful indication that the system needs to be changed is when users create their own forms or add additional information to existing ones. Such changes clearly demonstrate the need for improvements to existing systems. Thus, it is useful to review both blank and completed forms to identify these deviations. Likewise, when users access multiple reports to satisfy their information needs, it is a clear sign that new information or new information formats are needed.

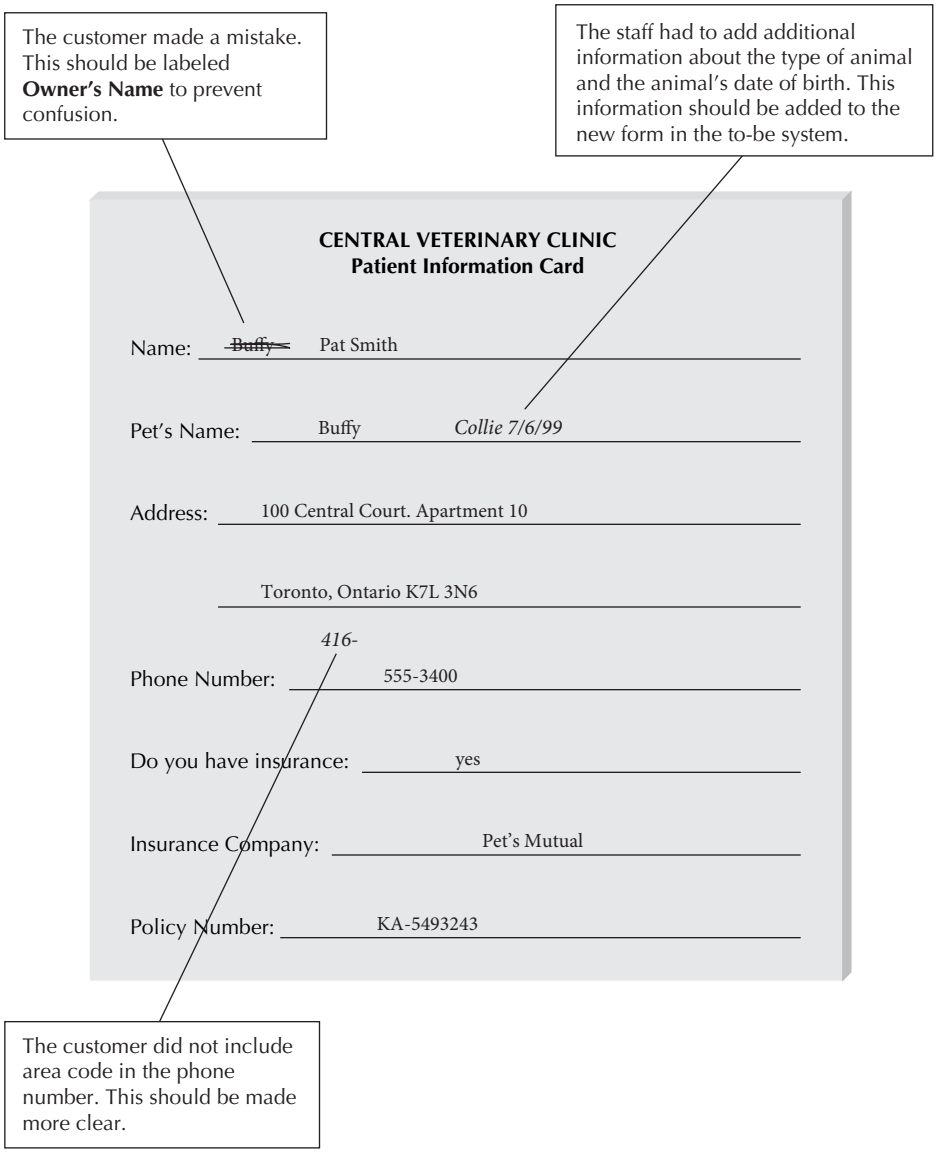


FIGURE 3-8
Performing a Document Analysis

Observation

Observation, the act of watching processes being performed, is a powerful tool for gathering information about the as-is system because it enables the analyst to see the reality of a situation, rather than listening to others describe it in interviews or JAD sessions. Several research studies have shown that many managers really do not remember how they work and how they allocate their time. (Quick, how many hours did you spend last week on each of your courses?) Observation is a good way to check the validity of information gathered from indirect sources such as interviews and questionnaires.

In many ways, the analyst becomes an anthropologist as he or she walks through the organization and observes the business system as it functions. The goal is to keep a low profile, to not interrupt those working, and to not influence those being observed. Nonetheless, it is important to understand that what analysts observe may not be the normal day-to-day routine because people tend to be extremely careful in their behavior when they are being watched. Even though normal practice may be to break formal organizational rules, the observer is unlikely to see this. (Remember how you drove the last time a police car followed you?) Thus, what you see might *not* be what you get.

Observation is often used to supplement interview information. The location of a person's office and its furnishings give clues to the person's power and influence in the organization and can be used to support or refute information given in an interview. For example, an analyst might become skeptical of someone who claims to use the existing computer system extensively if the computer is never turned on while the analyst visits. In most cases, observation supports the information that users provide in interviews. When it does not, it is an important signal that extra care must be taken in analyzing the business system.

Selecting the Appropriate Techniques

Each of the requirements-gathering techniques discussed earlier has strengths and weaknesses. No one technique is always better than the others, and in practice most projects use a combination of techniques. Thus, it is important to understand the strengths and weaknesses of each technique and when to use each (see Figure 3-9). One issue not discussed is that of the analysts' experience. In general, document analysis and observation require the least amount of training, whereas JAD sessions are the most challenging.

Type of Information The first characteristic is the type of information. Some techniques are more suited for use at different stages of the analysis process, whether understanding the as-is system, identifying improvements, or developing the to-be system. Interviews and JAD are commonly used in all three stages. In contrast, document analysis and observation usually are most helpful for understanding the as-is, although occasionally they provide information about

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low to Medium	Low	Low	Low to Medium

FIGURE 3-9 Table of Requirements-Gathering Techniques

current problems that need to be improved. Questionnaires are often used to gather information about the as-is system as well as general information about improvements.

Depth of Information The depth of information refers to how rich and detailed the information is that the technique usually produces and the extent to which the technique is useful for obtaining not only facts and opinions but also an understanding of *why* those facts and opinions exist. Interviews and JAD sessions are very useful for providing a good depth of rich and detailed information and helping the analyst to understand the reasons behind them. At the other extreme, document analysis and observation are useful for obtaining facts, but little beyond that. Questionnaires can provide a medium depth of information, soliciting both facts and opinions with little understanding of why they exist.

Breadth of Information Breadth of information refers to the range of information and information sources that can be easily collected using the chosen technique. Questionnaires and document analysis are both easily capable of soliciting a wide range of information from a large number of information sources. In contrast, interviews and observation require the analyst to visit each information source individually and, therefore, take more time. JAD sessions are in the middle because many information sources are brought together at the same time.

Integration of Information One of the most challenging aspects of requirements gathering is integrating the information from different sources. Simply put, different people can provide conflicting information. Combining this information and attempting to resolve differences in opinions or facts is usually very time consuming because it means contacting each information source in turn, explaining the discrepancy, and attempting to refine the information. In many cases, the individual wrongly perceives that the analyst is challenging his or her information, when in fact it is another user in the organization who is doing so. This can make the user defensive and make it hard to resolve the differences.

All techniques suffer integration problems to some degree, but JAD sessions are designed to improve integration because all information is integrated when it is collected, not afterward. If two users provide conflicting information, the conflict becomes immediately obvious, as does the source of the conflict. The immediate integration of information is the single most important benefit of JAD that distinguishes it from other techniques, and this is why most organizations use JAD for important projects.

User Involvement User involvement refers to the amount of time and energy the intended users of the new system must devote to the analysis process. It is generally agreed that as users become more involved in the analysis process, the chance of success increases. However, user involvement can have a significant cost, and not all users are willing to contribute valuable time and energy. Questionnaires, document analysis, and observation place the least burden on users, whereas JAD sessions require the greatest effort.

Cost Cost is always an important consideration. In general, questionnaires, document analysis, and observation are low-cost techniques (although observation can be quite time consuming). The low cost does not imply that they are more or less effective than the other techniques. Interviews and JAD sessions generally have moderate costs. In general, JAD sessions are much more expensive initially, because they require many users to be absent from their offices for significant periods of time, and they often involve highly paid consultants. However, JAD sessions significantly reduce the time spent in information integration and thus can cost less in the long term.

Combining Techniques In practice, requirements gathering combines a series of different techniques. Most analysts start by using interviews with senior manager(s) to gain an understanding of the project and the big-picture issues. From these interviews, it becomes clear whether large or small changes are anticipated. These interviews are often followed with analysis of documents

and policies to gain some understanding of the as-is system. Usually interviews come next to gather the rest of the information needed for the as-is picture.

In our experience, identifying improvements is most commonly done using JAD sessions because the JAD session enables the users and key stakeholders to work together through an analysis technique and come to a shared understanding of the possibilities for the to-be system. Occasionally, these JAD sessions are followed by questionnaires sent to a much wider set of users or potential users to see whether the opinions of those who participated in the JAD sessions are widely shared.

Developing the concept for the to-be system is often done through interviews with senior managers, followed by JAD sessions with users of all levels to make sure that the key needs of the new system are well understood.

ALTERNATIVE REQUIREMENTS DOCUMENTATION TECHNIQUES

Some other very useful requirements-gathering and documentation techniques include throwaway prototyping, use cases, role-playing CRC cards with use-case-based scenarios, concept mapping, and recording user stories on story cards and task lists. Throwaway prototyping was described in Chapter 1. In essence, throwaway prototypes are created to better understand some aspect of the new system. In many cases, they are used to test out some technical aspect of a nonfunctional requirement, such as connecting a client workstation to a server. If you have never done this before, it will be a lot easier to develop a very small example system to test out the necessary design of the connection from the client workstation to the server instead of trying to do it the first time with the full-blown system. Throwaway prototyping is very useful in designing user interfaces (see Chapter 10).

Use cases, as described in Chapter 1, are the fundamental approach that the Unified Process and Unified Modeling Language (UML) use to document and gather functional requirements. We describe them in Chapter 4. Role-playing CRC cards with use-case-based scenarios are very useful when creating functional (see Chapter 4), structural (see Chapter 5), and behavioral (see Chapter 6) models. We describe this approach in Chapter 5. The remainder of this section describes the use of concept mapping recording user stories on story cards and task lists.

Concept Maps

Concept maps represent meaningful relationships between concepts. They are useful for focusing individuals on the small number of key ideas on which they should concentrate. A concept map is essentially a node-and-arc representation, where the nodes represent the individual requirements and the arcs represent the relationships among the requirements. Each arc is labeled with a relationship name. Concept maps also have been recommended as a possible technique to support modeling requirements for object-oriented systems development and knowledge-management systems.[12] *Concept mapping* is an educational psychology technique that has been used in schools, corporations, and health care agencies to facilitate learning, understanding, and knowledge creation.[13] The advantage of the concept-mapping approach to representing requirements over the typical textual approach (see Figure 3-1) is that a concept map is not limited to a hierarchical representation. Concept maps allow the relationships among the functional and nonfunctional requirements to be explicitly represented. Figure 3-10 shows a concept map that portrays the information contained in the requirements

[12] See B. Henderson-Sellers, A. Simons, and H. Younessi, *The OPEN Toolbox of Techniques* (Harlow, England: Addison-Wesley, 1998).

[13] For more information on concept mapping, see J. D. Novak and D. B. Gowin, *Learning How to Learn* (Cambridge, UK: Cambridge University Press, 1984); J. D. Novak, *Learning, Creating, and Using Knowledge: Concept Maps™ as Facilitative Tools in Schools and Corporations* (Mahwah, NJ: Lawrence Erlbaum Associates, Publishers, 1998). Also, a free concept mapping tool is available from the Institute of Human and Machine Cognition at cmap.ihmc.us.

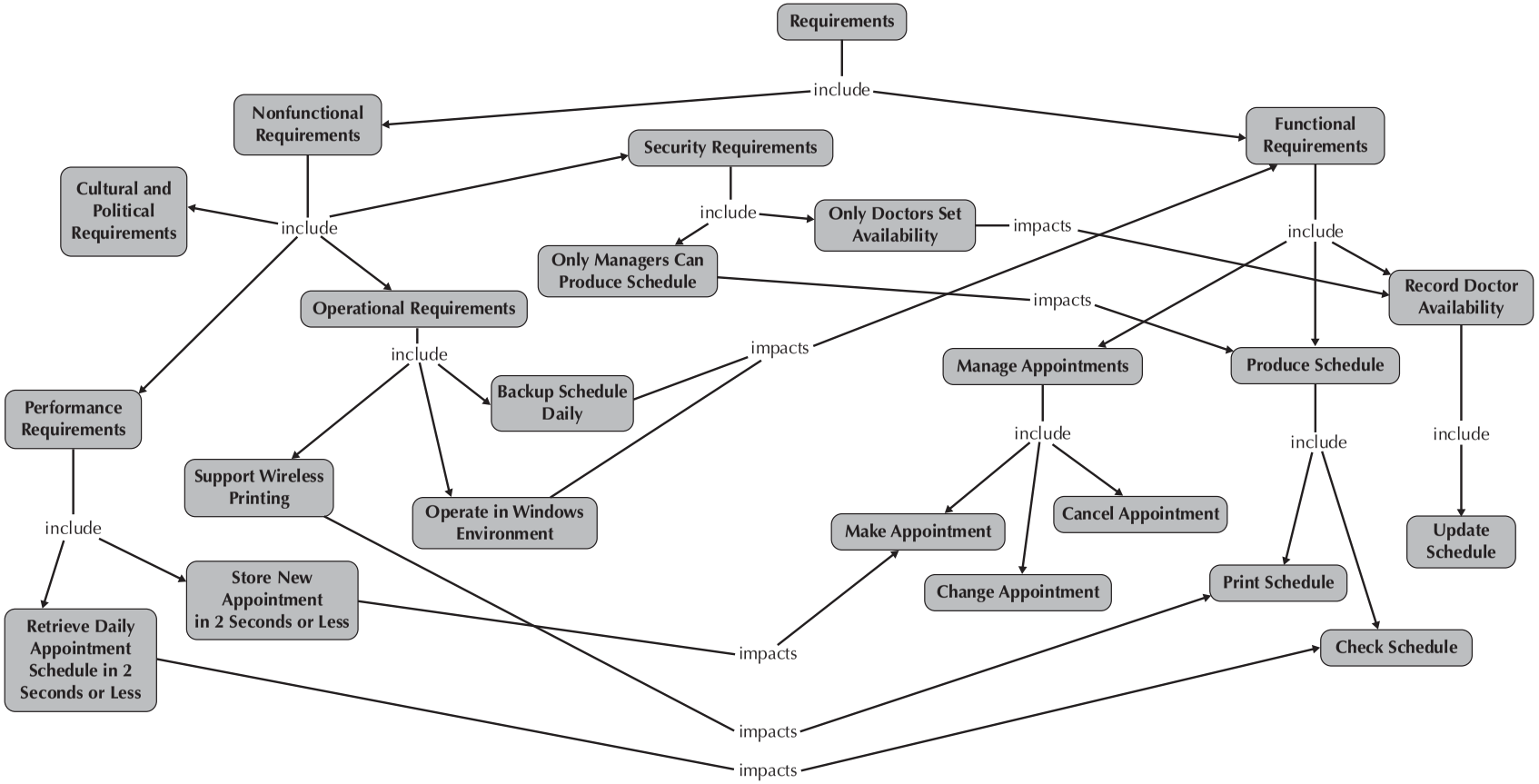


FIGURE 3-10 Sample Requirements Concept Map

definition shown in Figure 3-1. By using a concept map to represent the requirements instead of the textual approach, the relationship between the functional and nonfunctional requirements can be made explicit. For example, the two security requirements Only Doctors Set Availability and Only Managers Can Produce Schedule are explicitly linked to the Record Doctor Availability and Produce Schedule functional requirements, respectively. This is very difficult to represent in a text-only version of the requirements definition. Also, by having the user and analyst focus on the graphical layout of the map, additional requirements can be discovered. One obvious issue with this approach is that if the number of requirements becomes many and the relationships between them become complex, then the number of nodes and arcs will become so intertwined that the advantage of being able to explicitly see the relationships will be lost. However, by combining both text and concept-map representations, it is possible to leverage the strength of both textual and graphical representations to more completely represent the requirements.

User Stories

User stories, along with their associated *story cards* and *task lists*, are associated with the agile development approaches. User stories have been shown to be very useful in gathering requirements in a nonthreatening manner that respects the user's point of view. They are typically captured using story cards (index cards) and are recorded on a task list (or from a Scrum perspective, on the product backlog). Both story cards and task lists are considered to be lightweight approaches to documenting and gathering requirements.[14] Stories capture both functional and nonfunctional requirements. For example, with regard to the doctor's office appointment example, a functional requirement-based story could be:

> As a secretary, I want to be able to schedule appointments for our patients so that we can meet our patients' needs.

While an operational nonfunctional requirement-based story could be:

> As a secretary, I want to be able to print the daily schedule using wireless technology so that all printing can be performed using a shared printer without having to deal with printer cables connecting all of the computers to the printer.

Once the story is written down, it is discussed to determine the amount of effort it will take to implement it. During the discussion, a task list is created for the story. If the story is deemed to be too large—e.g., there are too many tasks on the task list—the story is split up into multiple stories each being recorded on its own story card and the tasks are allocated across the new stories. In many shops, once a set of tasks has been identified with a story, the story and its tasks are taped on a wall together so that all members of the development team can see the requirements. The story can be prioritized by importance by placing a rating on the card. The story can also be evaluated for the level of risk associated with it. The importance level and amount of risk associated with the story can be used to help choose which requirements to implement first. The advantage of using story cards and task lists to document requirements is that they are very low tech, high touch, easily updatable, and very portable.

[14] For more information on story cards and task lists see M. Cohn, *User Stories Applied: For Agile Software Development* (Boston, MA: Addison-Wesley, 2004); B. Rinzler, *Telling Stories: A Short Path to Writing Better Software Requirements* (Indianapolis, IN: Wiley, 2009); M. Lippert, S. Roock, H. Wolf, *eXtreme Programming in Action: Practical Experiences from Real World Projects* (Chichester, England: Wiley & Sons, Ltd., 2002); C. Larman, *Agile & Iterative Development: A Manager's Guide* (Boston, MA: Addison-Wesley, 2004).

THE SYSTEM PROPOSAL

A system proposal brings together into a single comprehensive document the material created during planning and analysis. The system proposal typically includes an executive summary, the system request, the workplan, the feasibility analysis, the requirements definition, and the evolving models that describe the new system. The evolving models include functional models (see Chapter 4), structural models (see Chapter 5), and behavioral models (see Chapter 6).[15] The executive summary provides all critical information in a very concise form. It can be thought of as a summary of the complete proposal. Its purpose is to allow a busy executive to quickly read through it and determine which parts of the proposal he or she needs to go through more thoroughly. The executive summary is typically no more than a single page long. Figure 3-11 provides a template for a system proposal and references to where the other sections of the proposal are described.

1. Table of Contents

2. Executive Summary

A summary of all the essential information in the proposal so that a busy executive can read it quickly and decide what parts of the proposal to read in more depth.

3. System Request

The revised system request form (see Chapter 2).

4. Workplan

The original workplan, revised after having completed analysis (see Chapter 2).

5. Feasibility Analysis

A revised feasibility analysis, using the information from analysis (see Chapter 2).

6. Requirements Definition

A list of the functional and nonfunctional business requirements for the system (this chapter).

7. Functional Model

An activity diagram, a set of use-case descriptions, and a use-case diagram that illustrate the basic processes or external functionality that the system needs to support (see Chapter 4).

8. Structural Models

A set of CRC cards, class diagram, and object diagrams that describe the structural aspects of the to-be system (see Chapter 5). This may also include structural models of the current as-is system that will be replaced.

9. Behavioral Models

A set of sequence diagrams, communication diagrams, behavioral-state machines, and a CRUDE matrix that describe the internal behavior of the to-be system (see Chapter 6). This may include behavioral models of the as-is system that will be replaced.

10. Appendices

These contain additional material relevant to the proposal, often used to support the recommended system. This might include results of a questionnaire survey or interviews, industry reports and statistics, and so on.

FIGURE 3-11
System Proposal Template

[15] Depending on the client, much more detailed specifications may be required; for example the Department of Defense, NASA, IEEE/ANSI, and the Naval Research Laboratory all have very specific formats that must be followed. For more information on these more detailed specifications, see A. M Davis, *Software Requirements, Revision* (Upper Saddle River, NJ: Prentice Hall, 1993); G. Kotonya and I. Sommerville, *Requirements Engineering* (Chichester, England: Wiley, 1998); R. H. Thayer and M. Dorfman (eds.), *Software Requirements Engineering,* 2nd Ed. (Los Alamitos, CA: IEEE Computer Society Press, 1997).