[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html" \l "header)

**Chapter**

**4**

**Requirements Engineering**

* [Chapter Introduction](javascript://)
* **4.1**[System Requirements](javascript://)
  + **4.1.1**[Types of Requirements](javascript://)
  + **4.1.2**[Requirements Challenges](javascript://)
  + **4.1.3**[Additional Considerations](javascript://)
* **4.2**[Team-Based Techniques](javascript://)
  + **4.2.1**[Joint Application Development](javascript://)
  + **4.2.2**[Rapid Application Development](javascript://)
  + **4.2.3**[Agile Methods](javascript://)
* **4.3**[Gathering Requirements](javascript://)
* **4.4**[Gathering Requirements through Interviews](javascript://)
  + **4.4.1**[The Interview Process](javascript://)
* **4.5**[Gathering Requirements Using Other Techniques](javascript://)
  + **4.5.1**[Document Review](javascript://)
  + **4.5.2**[Observation](javascript://)
  + **4.5.3**[Questionnaires and Surveys](javascript://)
  + **4.5.4**[Interviews Versus Questionnaires](javascript://)
  + **4.5.5**[Brainstorming](javascript://)
  + **4.5.6**[Sampling](javascript://)
  + **4.5.7**[Research](javascript://)
* **4.6**[Gathering Requirements in Agile Projects](javascript://)
* **4.7**[Representing Requirements](javascript://)
  + **4.7.1**[Natural Language](javascript://)
  + **4.7.2**[Diagrams](javascript://)
  + **4.7.3**[Models](javascript://)
* **4.8**[Validating and Verifying Requirements](javascript://)
* **4.9**[Tools](javascript://)
* **4.10**[Summary](javascript://)
* [Chapter Review](javascript://)
  + [Key Terms](javascript://)

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

# Chapter Introduction

**Chapter 4** is the first of the four chapters in the systems analysis phase. This chapter describes the requirements engineering process: gathering facts about a systems project, creating models that will be used to design and develop the system, and verifying and validating that the models are correct before proceeding to the next phase of the SDLC.

The chapter includes three “Case in Point” discussion questions to help contextualize the concepts described in the text. The “Question of Ethics” raises the issue of considering a request by a supervisor to identify departments that reported the lowest ratings in a survey that was supposed to be kept anonymous.

### Learning Objectives

When you finish this chapter, you should be able to:

1. Explain system requirements and the challenges associated with the requirements engineering process
2. Compare and contrast functional and non-functional requirements
3. Apply team-based requirements engineering techniques, including joint application development (JAD), rapid application development (RAD), and agile methods
4. Develop a fact-finding plan for gathering requirements
5. Conduct an interview to gather system requirements
6. Use other requirements gathering techniques, including document review, observation, questionnaires and surveys, brainstorming, sampling, and research
7. Explain how requirements are gathered in agile projects
8. Utilize different requirements representation techniques, including natural language, diagrams, and models
9. Explain how to validate and verify requirements
10. Explain how tools can help with requirements engineering activities

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.1**System Requirements

During the first part of the systems analysis phase of the SDLC, systems analysts must identify and describe all system requirements. A [**system requirement**](javascript://) is a characteristic or feature that must be included in an information system to satisfy business requirements and be acceptable to users. System requirements serve as benchmarks to measure the overall acceptability of the finished system.

Because system requirements are the focus of the start of the systems analysis phase, it’s important to get them correct, right at the start of the process. Any problems with the requirements will have a ripple effect that could negatively affect subsequent phases of the SDLC. In fact, poor **requirements engineering** is a leading cause of failed projects.

Requirements engineering is composed of three main activities:

1. Gathering requirements: *understanding* the problem
2. Representing requirements: *describing* the problem
3. Validating and verifying requirements: *agreeing* upon the problem

Each of these activities is described in more detail in this chapter. The output of requirements engineering are requirements documents that capture the essence of what the system should do. These documents are the input to the next step in the SDLC, data and process modeling, which is described in [Chapter 5](javascript://).

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.1.1Types of Requirements

Requirements can be classified according to various characteristics. For example, requirements may be primarily for the system user, in which case they are referred to as [**requirements definitions**](javascript://). Requirements may also be primarily for the engineering team, in which case they are referred to as [**requirements specifications**](javascript://).

System requirements can also be classified as functional and non-functional. A [**functional requirement**](javascript://) is a statement of the services a system provides. Examples of functional requirements include the following:

* The website shall report online volume statistics every four hours and hourly during peak periods.
* The inventory system shall produce a daily report showing the part number, description, quantity on hand, quantity allocated, quantity available, and unit cost of all sorted by part number.
* The contact management system shall generate a daily reminder list for all sales representatives.
* Each input form must include date, time, product code, customer number, and quantity.
* The system must provide logon security at the operating system level and at the application level.

A [**non-functional requirement**](javascript://) is a statement of operational system constraints. Non-functional requirements are also known as [**quality attributes**](javascript://). Examples of non-functional requirements include the following:

* Data entry screens must be uniform, except for background color, which can be changed by the user.
* The system must support 25 users online simultaneously.
* Response time must not exceed four seconds.
* The system must be operational 7 days a week, 365 days a year.
* The system should work on Windows and Mac platforms.

Non-functional requirements may be more critical than functional requirements; if the former are not satisfied, the system is useless. Conflicts between different non-functional requirements are common in complex systems. For example, a user may request that a system be 100% secure but very usable—two requirements that are hard to reconcile.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.1.2Requirements Challenges

Requirements present numerous challenges to the systems analyst. Three of the most important are imprecision, agreement, and creep.

### Imprecision

Requirements are often imprecise because they are usually represented using natural language, such as the examples shown in [Section 4.1.1](javascript://). Natural language is expressive, but it is prone to misinterpretation. It is not uncommon for various stakeholders to completely disagree as to the meaning of a simple requirement. It is for this reason that other techniques are used to represent requirements, as described in [Section 4.7](javascript://).

A requirement may be a high-level abstract statement of a service or of a system constraint, but it can also be a detailed mathematical specification. This is because requirements often serve two functions: as a basis for a bid for a contract, and as the basis for the contract itself.

If the requirements are the basis for the contract bid, it must be open to interpretation. But if the requirements are the basis for the contract itself, it must be defined in detail. These two constraints are difficult to satisfy.

### Agreement

One of the main problems with requirements is getting everyone to agree on the exact meaning of the requirements statements. In other words, we want to develop requirements in such a way that, upon completion of the system, both the systems analyst and the client can agree on whether or not a specific requirement has been met.

In theory, requirements should be both complete and consistent. A requirement is complete if it includes descriptions of all facilities needed by the system. In practice, it’s impossible to completely describe the requirements for a complex system.

A requirement is consistent if there are no conflicts or contradictions in the description of the system facilities. In practice, ensuring that there are no conflicts in the system requirements when there may be thousands is quite challenging.

### Creep

There are many social and organizational factors that influence system requirements. For example, business changes inevitably lead to changing requirements—usually more of them as the project progresses. This is particularly true for long-lived projects where the personnel involved change over time. This phenomenon is known as “feature creep.”

Rapidly changing requirements can cause numerous problems for systems analysts and other team members. This is particularly true for projects that follow a traditional waterfall model of the SDLC. It is partly for this reason that agile methods are popular: They explicitly address changing requirements as part of the project’s management structure.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.1.3Additional Considerations

In addition to the three challenges outlined earlier, systems analysts must consider three important supplementary factors:

* (1)

scalability, which determines how a system will handle future growth and demands;

* (2)

security, which is all-important for today’s networked systems; and

* (3)

the total cost of ownership, which includes all future operational and support costs.

### Scalability

[**Scalability**](javascript://) refers to a system’s ability to handle increased business volume and transactions in the future. Because it will have a longer useful life, a scalable system offers a better return on the initial investment.

To evaluate scalability, information is needed about projected future volume for all outputs, inputs, and processes. For example, for a web-based order processing system, one needs to know the maximum projected number of concurrent users, the periods of peak online activity, the number and types of data items required for each transaction, and the method of accessing and updating customer files.

Even to print customer statements, the analyst needs to know the number of active accounts and have a forecast for one, two, or five years because that information affects future hardware decisions. In addition, with realistic volume projections, reliable cost estimates for related expenses, such as postage and online charges, can be provided.

Similarly, to ensure that a web-based hotel reservation system is sufficiently scalable, the analyst would need to project activity levels for several years of operation. For example, one might forecast the frequency of online queries about room availability and estimate the time required for each query and the average response time. With that information, server transaction volume and network requirements could be estimated.

Transaction volume has a significant impact on operating costs. When volume exceeds a system’s limitations, maintenance costs increase sharply. Volume can change dramatically if a company expands or enters a new line of business. For example, a new Internet-based marketing effort might require an additional server and a 24-hour technical support.

Data storage also is an important scalability issue. The analyst must determine how much data storage is required currently and predict future needs based on system activity and growth. Those requirements affect hardware, software, and network bandwidth needed to maintain system performance. Data retention requirements must also be considered to determine whether data can be deleted or archived on a specific timetable.

### Security

In the past, security was considered an add-on to a system’s design. This is particularly true for legacy systems that have been deployed for some time. Nowadays, security is so important for networked systems that it has changed from a non-functional requirement to a functional requirement. In other words, security is an essential consideration for all systems development.

Incorporating security as a first-class requirement is particularly important in light of the seemingly endless news reports of massive data breaches. These hacks release personal information from online systems at a scale previously unseen. If the systems had been made with security in mind from the beginning, they would be much harder to infiltrate.

The challenge with security as a system requirement is that it is often in conflict with other user requirements. For example, a requirement that the system should be accessible online using a web interface immediately makes securing the system much more challenging. The systems analyst must attempt to reconcile and prioritize the conflicting requirements during the requirements engineering process.

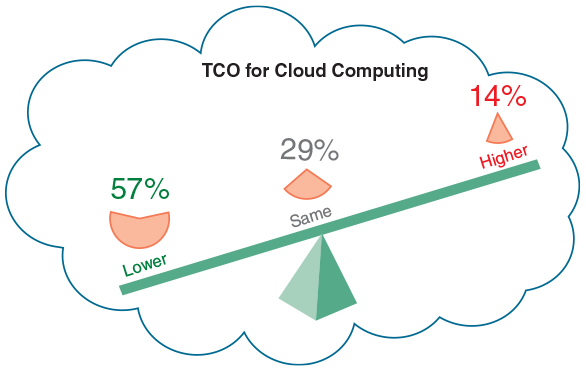
### Total Cost of Ownership

In addition to direct costs, systems developers must identify and document indirect expenses that contribute to the **total cost of ownership (TCO)**. TCO is especially important if the development team is assessing several alternatives. After considering the indirect costs, which are not always apparent, a system that seems inexpensive initially might actually turn out to be the costliest choice. One problem is that cost estimates tend to understate indirect costs such as user support and downtime productivity losses. Even if accurate figures are unavailable, systems analysts should try to identify indirect costs and include them in TCO estimates.

Because cost control is so important, vendors often claim that their products or services will reduce TCO significantly. For example, one of the most common reasons to migrate a legacy system to the cloud is reduced TCO. As shown in [Figure 4-1](javascript://), cloud computing offers the opportunity for lower operational costs due to the outsourcing of expenses such as capital investment in exchange for a pay-as-you-go pricing model.

**Figure 4-1**

Total cost of ownership when migrating to the cloud can be significantly less than current computing platforms.



Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.2**Team-Based Techniques

The IT department’s goal is to deliver the best possible information system, at the lowest possible cost, in the shortest possible time. To achieve the best results, systems developers view users as partners in the development process. Greater user involvement usually results in better communication, faster development times, and more satisfied users.

The traditional model for systems development was an IT department that used structured analysis and consulted users only when their input or approval was needed. Although the IT staff still has a central role, and structured analysis remains a popular method of systems development, most IT managers invite system users to participate actively in various development tasks.

As described in [Chapter 1](javascript://), team-based approaches have been around for some time. A popular example is joint application development (JAD), which is a user-oriented technique for fact-finding and requirements engineering. Because it is not linked to a specific development methodology, systems developers use JAD whenever group input and interaction are desired.

Another popular user-oriented method is rapid application development (RAD). RAD resembles a condensed version of the entire SDLC, with users involved every step of the way. While JAD typically focuses only on fact-finding and requirements determination, RAD provides a fast-track approach to a full spectrum of systems development tasks, including planning, design, construction, and implementation.

Finally, as described in [Chapter 1](javascript://), agile methods represent a recent trend that stresses intense interaction between systems developers and users. JAD, RAD, and agile methods are discussed in the following sections.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.2.1Joint Application Development

[**Joint application development (JAD)**](javascript://) is a popular fact-finding technique that brings users into the development process as active participants.

### User Involvement

Users have a vital stake in an information system, and they should participate fully in the development process. Many years ago, the IT department usually had sole responsibility for systems development and users had a relatively passive role. During the development process, the IT staff would collect information from users, define system requirements, and construct the new system. At various stages of the process, the IT staff might ask users to review the design, offer comments, and submit changes.

Today, users typically have a much more active role in systems development. IT professionals now recognize that successful systems must be user oriented, and users need to be involved, formally or informally, at every stage of systems development.

One popular strategy for user involvement is a JAD team approach, which involves a task force of users, managers, and IT professionals who work together to gather information, discuss business needs, and define the new system requirements.

### JAD Participants and Roles

A JAD team usually meets over a period of days or weeks in a special conference room or at an off-site location. Either way, JAD participants should be insulated from the distraction of day-to-day operations. The objective is to analyze the existing system, obtain user input and expectations, and document user requirements for the new system.

The JAD group usually has a project leader, who needs strong interpersonal and organizational skills, and one or more members who document and record the results and decisions. [Figure 4-2](javascript://) describes typical JAD participants and their roles. IT staff members often serve as JAD project leaders, but that is not always the case. Systems analysts on the JAD team participate in discussions, ask questions, take notes, and provide support to the team. If CASE tools are available, analysts can develop models and enter documentation from the JAD session directly into the CASE tool.

**Figure 4-2**

Typical JAD participants and roles.

| **JAD PARTICIPANT** | **ROLE** |
| --- | --- |
| JAD project leader | Develops an agenda, acts as a facilitator, and leads the JAD session |
| Top management | Provides enterprise-level authorization and support for the project |
| Managers | Provide department-level support for the project and understanding of how the project must support business functions and requirements |
| Users | Provide operational-level input on current operations, desired changes, input and output requirements, user interface issues, and how the project will support day-to-day tasks |
| Systems analysts and other IT staff members | Provide technical assistance and resources for JAD team members on issues such as security, backup, hardware, software, and network capability |
| Recorder | Documents results of JAD sessions and works with systems analysts to build system models and develop CASE tool documentation |

Enlarge Table

A typical JAD session agenda is shown in [Figure 4-3](javascript://). The JAD process involves intensive effort by all team members. Because of the wide range of input and constant interaction among the participants, many companies believe that a JAD group produces the best possible definition of the new system.

**Figure 4-3**

Typical agenda for a JAD session.

|  |  |
| --- | --- |
| Project leader | * Introduce all JAD team members * Discuss ground rules, goals, and objectives for the JAD sessions * Explain methods of documentation and use of CASE tools, if any |
| Top management (sometimes called the project owner or sponsor) | * Explain the reason for the project and express top management authorization and support |
| Project leader | * Provide overview of the current system and proposed project scope and constraints * Present outline of specific topics and issues to be investigated |
| Open discussion session, moderated by project leader | * Review the main business processes, tasks, user roles, input, and output * Identify specific areas of agreement or disagreement * Break team into smaller groups to study specific issues and assign group leaders |
| JAD team members working in smaller group sessions, supported by IT staff | * Discuss and document all system requirements * Develop models and prototypes |
| Group leaders | * Report on results and assigned tasks and topics * Present issues that should be addressed by the overall JAD team |
| Open discussion session, moderated by project leader | * Review reports from small group sessions * Reach consensus on main issues * Document all topics |
| Project leader | * Present overall recap of JAD session * Prepare report that will be sent to JAD team members |

Enlarge Table

### JAD Advantages and Disadvantages

Compared with traditional methods, JAD is more expensive and can be cumbersome if the group is too large relative to the size of the project. Many companies find, however, that JAD allows key users to participate effectively in the requirements engineering process. When users participate in the systems development process, they are more likely to feel a sense of ownership in the results and support for the new system. When properly used, JAD can result in a more accurate statement of system requirements, a better understanding of common goals, and a stronger commitment to the success of the new system.

**Case in Point 4.1**

### North Hills College

* North Hills College has decided to implement a new registration system that will allow students to register online as well as in person. As IT manager, you decide to set up a JAD session to help define the requirements for the new system. The North Hills organization is fairly typical, with administrative staff that includes a registrar, a student support and services team, a business office, an IT group, and a number of academic departments. Using this information, you start work on a plan to carry out the JAD session. Who would you invite to the session, and why? What would be your agenda for the session, and what would take place at each stage of the session?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.2.2Rapid Application Development

[**Rapid application development (RAD)**](javascript://) is a team-based technique that speeds up information systems development and produces a functioning information system. Like JAD, RAD uses a group approach but goes much further. While the end product of JAD is a requirements model, the end product of RAD is the new information system. RAD is a complete methodology, with a four-phase life cycle that parallels the traditional SDLC phases. Companies use RAD to reduce cost and development time and increase the probability of success.

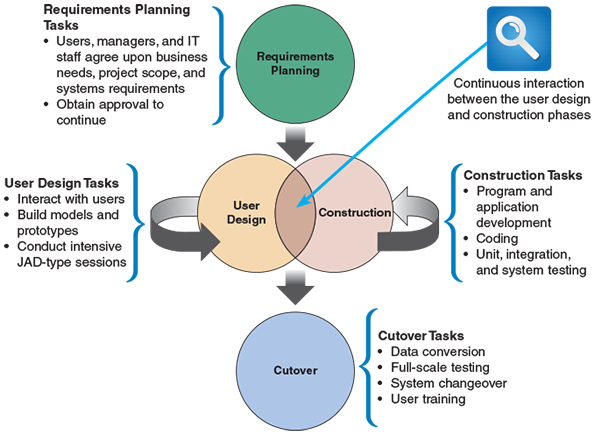
RAD relies heavily on prototyping and user involvement. The RAD process allows users to examine a working model as early as possible, determine if it meets their needs, and suggest necessary changes. Based on user input, the prototype is modified, and the interactive process continues until the system is completely developed and users are satisfied. The project team uses CASE tools to build the prototypes and create a continuous stream of documentation.

### RAD Phases and Activities

The RAD model consists of four phases: requirements planning, user design, construction, and cutover, as shown in [Figure 4-4](javascript://). Note the continuous interaction between the user design and construction phases.

**Figure 4-4**

The four phases of the RAD model are requirements planning, user design, construction, and cutover. Note the continuous interaction between the user design and construction phases.



Enlarge Image

#### Requirements Planning

The [**requirements planning phase**](javascript://) combines elements of the systems planning and systems analysis phases of the SDLC. Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements. The requirements planning phase ends when the team agrees on the key issues and obtains management authorization to continue.

#### User Design

During the [**user design phase**](javascript://), users interact with systems analysts and develop models and prototypes that represent all system processes, outputs, and inputs. The RAD group or subgroups typically use a combination of JAD techniques and CASE tools to translate user needs into working models. User design is a continuous, interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.

#### Construction

The [**construction phase**](javascript://) focuses on program and application development tasks similar to the SDLC. In RAD, however, users continue to participate and still can suggest changes or improvements as actual screens or reports are developed.

#### Cutover

The [**cutover phase**](javascript://) resembles the final tasks in the SDLC implementation phase, including data conversion, testing, changeover to the new system, and user training. Compared with traditional methods, the entire process is compressed. As a result, the new system is built, delivered, and placed in operation much sooner.

### RAD Objectives

The main objective of all RAD approaches is to cut development time and expense by involving users in every phase of systems development. Because it is a continuous process, RAD allows the development team to make necessary modifications quickly, as the design evolves. In times of tight corporate budgets, it is especially important to limit the cost of changes that typically occur in a long, drawn-out development schedule.

In addition to user involvement, a successful RAD team must have IT resources, skills, and management support. Because it is a dynamic, user-driven process, RAD is especially valuable when a company needs an information system to support a new business function. By obtaining user input from the beginning, RAD also helps a development team design a system that requires a highly interactive or complex user interface.

### RAD Advantages and Disadvantages

RAD has advantages and disadvantages compared with traditional structured analysis methods. The primary advantage is that systems can be developed more quickly with significant cost savings. A disadvantage is that RAD stresses the mechanics of the systems itself and does not emphasize the company’s strategic business needs. The risk is that a system might work well in the short term, but the corporate and long-term objectives for the system might not be met. Another potential disadvantage is that the accelerated time cycle might allow less time to develop quality, consistency, and design standards. RAD can be an attractive alternative, however, if an organization understands the possible risks.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.2.3Agile Methods

[Chapter 1](javascript://) explained that **agile methods** attempt to develop a system incrementally by building a series of prototypes and constantly adjusting them to user requirements. As the agile process continues, developers revise, extend, and merge earlier versions into the final product. An agile approach emphasizes continuous feedback, and each incremental step is affected by what was learned in the prior steps.

As agile methods become more popular, a large community of agile-related software and services has evolved. Many agile developers prefer not to use CASE tools at all, and as shown in [Figure 4-5](javascript://) rely instead on whiteboard displays and arrangements of movable sticky notes. This approach, they believe, reinforces the agile strategy: simple, rapid, flexible, and user oriented.

**Figure 4-5**

Reinforcing the agile strategy: simple, rapid, flexible, and user oriented.



Enlarge Image

Sam Edwards/OJO Images/Getty Images

[**Scrum**](javascript://) is another agile approach. The name comes from the rugby term scrum, where team members lunge at each other to achieve their objectives, as shown in [Figure 4-6](javascript://). The systems development version of Scrum involves the same intense interaction, though it is more mental than physical. In a Scrum session, agile team members play specific roles, including colorful designations such as pigs or chickens. These roles are based on the old joke about the pig and chicken who discuss a restaurant where ham and eggs would be served. However, the pig declines, because that role would require a total commitment, while for the chicken, it would only be a contribution.

**Figure 4-6**

In a rugby scrum, team members prepare to lunge at each other to achieve their objectives.



Enlarge Image

getnikov/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

In the agile world, the pigs include the product owner, the facilitator, and the development team, while the chickens include users, other stakeholders, and managers. Scrum sessions have specific guidelines that emphasize time blocks, interaction, and team-based activities that result in deliverable software. An agile team uses a series of scrums to pause the action and allow the players to reset the game plan, which remains in effect until the next scrum.

### Agile Method Advantages and Disadvantages

Agile, or adaptive, methods are very flexible and efficient in dealing with change. They are popular because they stress team interaction and reflect a set of community-based values. Also, frequent deliverables constantly validate the project and reduce risk.

However, some potential problems exist. For example, team members need a high level of technical and interpersonal skills. Also, a lack of structure and documentation can introduce risk factors, such as blurring of roles and responsibilities, and loss of corporate knowledge. Finally, the overall project may be subject to significant change in scope as user requirements continue to evolve during the project.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.3**Gathering Requirements

Gathering requirements is the first step in the requirements engineering process. This step is also known as [**requirements elicitation**](javascript://) or [**fact-finding**](javascript://) (collecting information). Whether working solo or as a member of a team, during requirements gathering, the systems analyst will use various techniques, including interviews, document review, observation, surveys and questionnaires, sampling, and research.

Although software can help gather and analyze requirements, no program actually gathers them automatically. First, the information needed must be identified. Typically, this activity begins by asking a series of questions, such as the following:

* What business functions are supported by the current system?
* What strategic objectives and business requirements must be supported by the new system?
* What are the benefits and TCO of the proposed system?
* What transactions will the system process?
* What information do users and managers need from the system?
* Must the new system interface with legacy systems?
* What procedures could be eliminated by business process reengineering?
* What security issues exist?
* What risks are acceptable?
* What budget and timetable constraints will affect systems development?

To obtain answers to these questions, the analyst develops a fact-finding plan, which involves answers to five familiar questions: who, what, where, when, and how. For each of those questions, one also must ask another very important question: why. Some examples of these questions are as follows:

1. *Who*? Who performs each of the procedures within the system? Why? Are the correct people performing the activity? Could other people perform the tasks more effectively?
2. *What*? What is being done? What procedures are being followed? Why is the process necessary? Often, procedures are followed for many years and no one knows why. Question why a procedure is being followed at all.
3. *Where*? Where are operations being performed? Why? Where could they be performed? Could they be performed more efficiently elsewhere?
4. *When*? When is a procedure performed? Why is it being performed at this time? Is this the best time?
5. *How*? How is a procedure performed? Why is it performed in that manner? Could it be performed better, more efficiently, or less expensively in some other manner?

There is a difference between asking what is being done and what could or should be done. The systems analyst first must understand the current situation. Only then can the question of what should be done be answered. [Figure 4-7](javascript://) lists the basic questions and when they should be asked. Note that the first two columns relate to the current system but the third column focuses on the proposed system.

**Figure 4-7**

Sample questions during requirements elicitation as the focus shifts from the current system to the proposed system.

| **CURRENT SYSTEM** | | **PROPOSED SYSTEM** |
| --- | --- | --- |
| Who does it? | Why does this person do it? | Who should do it? |
| What is done? | Why is it done? | What should be done? |
| Where is it done? | Why is it done there? | Where should it be done? |
| When is it done? | Why is it done then? | When should it be done? |
| How is it done? | Why is it done this way? | How should it be done? |

Enlarge Table

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.4**Gathering Requirements through Interviews

Interviewing is an important requirement gathering technique during the systems analysis phase. An [**interview**](javascript://) is a planned meeting during which the analyst obtains information from another person. The skills needed to plan, conduct, document, and evaluate interviews successfully must be understood.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.4.1The Interview Process

After identifying the information needed, as described earlier in the chapter, the interviewing process commences, which consists of seven steps for each interview:

1. Determine the people to interview.
2. Establish objectives for the interview.
3. Develop interview questions.
4. Prepare for the interview.
5. Conduct the interview.
6. Document the interview.
7. Evaluate the interview.

### Step 1: Determine the People to Interview

To get an accurate picture, the analyst must select the right people to interview and ask them the right questions. The preliminary investigation involved mainly middle managers or department heads. Now, during the systems analysis phase, people from all levels of the organization should be interviewed. In some situations, it might be prudent to interview stakeholders that are not members of the organization, because their opinion is valuable.

Although interview candidates can be selected from the formal organization charts reviewed earlier, one must also consider any informal structures that exist in the organization. [**Informal structures**](javascript://) usually are based on interpersonal relationships and can develop from previous work assignments, physical proximity, unofficial procedures, or personal relationships such as the informal gathering shown in [Figure 4-8](javascript://). In an informal structure, some people have more influence or knowledge than appears on an organization chart. The analyst’s knowledge of the company’s formal and informal structures helps determine the people to interview during the systems analysis phase.

**Figure 4-8**

When setting up interviews, an analyst should look outside a formal organization chart to identify people who might provide valuable information.



GaudiLab/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

Should several people be interviewed at the same time? Group interviews can save time and provide an opportunity to observe interaction among the participants. Group interviews also can present problems. One person might dominate the conversation, even when questions are addressed specifically to others. Organization level also can present a problem because the presence of senior managers in an interview might prevent lower-level employees from expressing themselves candidly.

### Step 2: Establish Objectives for the Interview

After deciding on the people to interview, objectives for the session must be established. First, the general areas to be discussed should be determined, and then the facts to be gathered should be listed. Soliciting ideas, suggestions, and opinions during the interview is also a good idea.

The objectives of an interview depend on the role of the person being interviewed. Upper-level managers can provide the big picture to help understand the system as a whole. Specific details about operations and business processes are best learned from people who actually work with the system on a daily basis.

In the early stages of systems analysis, interviews usually are general. As the fact-finding process continues, however, the interviews focus more on specific topics. Interview objectives also vary at different stages of the investigation. Interviews should be as brief as possible (though as long as needed), since time is so valuable to employees, especially managers. By setting specific objectives, a framework is created that helps the analyst decide what questions to ask and how to phrase them.

### Step 3: Develop Interview Questions

Creating a standard list of interview questions helps to keep the session on track and avoid unnecessary tangents. Also, if several people who perform the same job are interviewed, a standard question list permits a comparison of their answers. Although there may be a list of specific questions, the interviewer might decide to depart from it because an answer to one question leads to another topic that warrants pursuing. That question or topic then should be included in a revised set of questions used to conduct future interviews. If the question proves to be extremely important, it may be needed to return to a previous interviewee to query him or her on the topic.

The interview should consist of several different kinds of questions: open-ended, closed-ended, or questions with a range of responses. When phrasing questions, avoid [**leading questions**](javascript://) that suggest or favor a particular reply. For example, rather than asking, “What advantages do you see in the proposed system?” ask instead, “Do you see any advantages in the proposed system?”

#### Open-Ended Questions

[**Open-ended questions**](javascript://) encourage spontaneous and unstructured responses. Such questions are useful to understand a larger process or draw out the interviewee’s opinions, attitudes, or suggestions. Here are some examples of open-ended questions:

* What are users saying about the new system?
* How is this task performed?
* Why do you perform the task that way?
* How are the checks reconciled?
* What added features would you like to have in the new billing system?

Also, an open-ended question can be used to probe further by asking: Is there anything else you can tell me about this topic?

#### Closed-Ended Questions

[**Closed-ended questions**](javascript://) limit or restrict the response. Closed-ended questions are used when information that is more specific is needed or when facts must be verified. Examples of closed-ended questions include the following:

* How many personal computers do you have in this department?
* Do you review the reports before they are sent out?
* How many hours of training does a clerk receive?
* Is the calculation procedure described in the manual?
* How many customers ordered products from the website last month?

#### Range-Of-Response Questions

[**Range-of-response questions**](javascript://) are closed-ended questions that ask the person to evaluate something by providing limited answers to specific responses or on a numeric scale. This method makes it easier to tabulate the answers and interpret the results. Range-of-response questions might include the following:

* On a scale of 1 to 10, with 1 the lowest and 10 the highest, how effective was your training?
* How would you rate the severity of the problem: low, medium, or high?
* Is the system shutdown something that occurs never, sometimes, often, usually, or always?

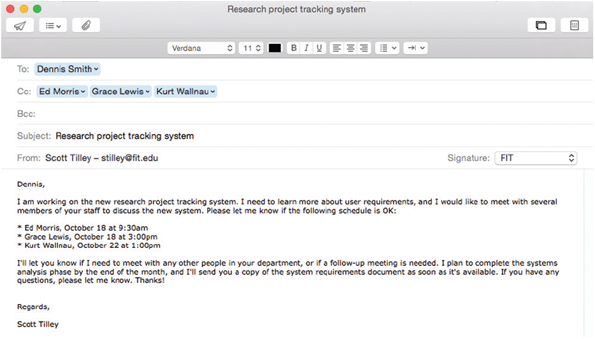
### Step 4: Prepare for the Interview

After setting the objectives and developing the questions, preparing for the interview is next. Careful preparation is essential because an interview is an important meeting and not just a casual chat. When the interview is scheduled, suggest a specific day and time and let the interviewee know how long the meeting is expected to last. It is also a good idea to send an email or place a reminder call the day before the interview.

Remember that the interview is an interruption of the other person’s routine, so the interview should be limited to no more than one hour. If business pressures force a postponement of the meeting, schedule another appointment as soon as it is convenient. Remember to keep department managers informed of meetings with their staff members. Sending a message to each department manager listing planned appointments is a good way to keep them informed. [Figure 4-9](javascript://) is an example of such a message.

**Figure 4-9**

Sample message to a department head about interviews.

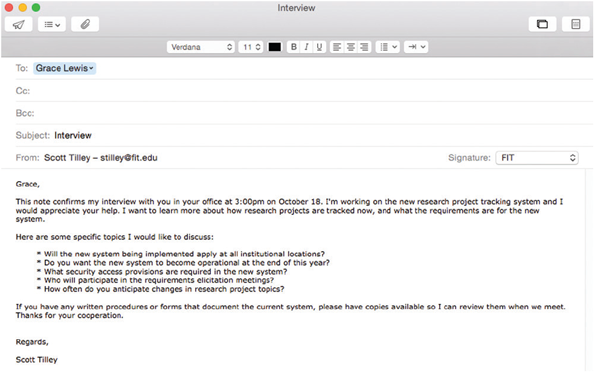


Enlarge Image

A list of topics should be sent to an interviewee several days before the meeting, especially when detailed information is needed, so the person can prepare for the interview and minimize the need for a follow-up meeting. [Figure 4-10](javascript://) shows a sample message that lists specific questions and confirms the date, time, location, purpose, and anticipated duration of the interview.

**Figure 4-10**

Sample message to confirm an interview.



Enlarge Image

If there are questions about documents, ask the interviewee to have samples available at the meeting. The advance memo should include a list of the documents to discuss (if it is known what they are). Also, a general request for documents can be made, as the analyst did in the email shown in [Figure 4-10](javascript://).

Two schools of thought exist about the best location for an interview. Some analysts believe that interviews should take place in the interviewee’s office, whereas other analysts feel that a neutral location such as a conference room is better.

Supporters of interviews in the interviewee’s office believe that is the best location because it makes the interviewee feel comfortable during the meeting. A second argument in favor of the interviewee’s office is that the office is where he or she has the easiest access to supporting material that might be needed during the discussion. If a complete list of topics is provided in advance, however, the interviewee can bring the necessary items to a conference room or other location.

Supporters of neutral locations stress the importance of keeping interruptions to a minimum so both people can concentrate fully. In addition, an interview that is free of interruptions takes less time. If the meeting does take place in the interviewee’s office, tactfully suggest that all calls be held until the conclusion of the interview.

### Step 5: Conduct the Interview

After determining the people to interview, setting the objectives, and preparing the questions, a specific plan for the meeting should be developed. When conducting an interview, begin with introductions, describe the project, and explain the interview objectives.

During the interview, ask questions in the order in which they were prepared and give the interviewee sufficient time to provide thoughtful answers. Some answers will lead to additional questions, which should be asked in a logical order. Establishing a good rapport with the interviewee is important, especially if this is the first meeting. If the other person feels comfortable and at ease, they will probably provide more complete and candid answers. The analyst’s primary responsibility during an interview is to listen carefully to the answers. Analysts sometimes hear only what they expect to hear. Concentrate on what is said and notice any nonverbal communication that takes place. This process is called [**engaged listening**](javascript://).

After asking a question, allow the person enough time to think about the question and arrive at an answer. Studies have shown that the maximum pause during a conversation is usually three to five seconds. After that interval, one person will begin talking. An analyst needs to be patient and practice his or her skills in many actual interview situations to be successful.

When all the questions have been asked, summarize the main points covered in the interview and explain the next course of action. For example, mention that a follow-up memo will be sent or that the interviewee should send certain requested information after the meeting. When the interview has concluded, thank the person and encourage him or her to reach out with any questions or additional comments. Also, when the interview ends, it is a good idea to ask the interviewee whether he or she can suggest any additional topics that should be discussed.

After an interview, summarize the session and seek a confirmation from the other person. By stating the interviewer’s understanding of the discussion, the interviewee can respond and provide corrections, if necessary. One approach is to rephrase the interviewee’s answers. For example, the analyst could say, “If I understand you correctly, you are saying that . . . .” and then reiterate the information given by the interviewee.

### Step 6: Document the Interview

Although taking notes during an interview has both advantages and disadvantages, it should be kept to a minimum. It is a good idea to write down a few notes to remember key points after the interview but avoid writing down everything that is said. Too much writing distracts the other person and makes it harder to establish a good rapport.

After conducting the interview, record the information quickly. Set aside time right after the meeting to record the facts and evaluate the information. For that reason, try not to schedule back-to-back interviews. Studies have shown that 50% of a conversation is forgotten within 30 minutes. Therefore, use the notes to record the facts immediately so they will not be forgotten. Summarize the facts by preparing a narrative describing what took place or by recording the answers received next to each question on the prepared list.

Small, portable recorders are effective tools for interviews, but some people are uncomfortable when they are used. Before using a recorder, discuss its use with the interviewee. Assure the interviewee that the recording will be erased after its contents are transcribed into note form and that the interview can be stopped at any time at the interviewee’s request. If sensitive questions are asked, or the interviewee wants to answer a question without being recorded, explain that the recorder can be turned off for a period of time during the interview.

Instead of using a traditional recorder that calls attention to its presence, an interviewer can use built-in audio (or even video) recording features on a notebook or mobile device. Also, as pointed out in [Section 4.9](javascript://), an interviewer can use powerful information management software, such as Microsoft OneNote, to record the meeting, store the results, and create a searchable file for easy access. Irrespective of the mechanism used to record the meeting, all participants should be aware that what they say is being recorded.

Whether or not the meeting is recorded, listen carefully to the interviewee’s responses so good follow-up questions can be asked. Otherwise, a second visit might be needed to ask the questions missed the first time. Also, remember that each recorded interview takes twice the amount of time, because the analyst must listen to or view the recorded meeting again after conducting the interview itself.

After the meeting, a memo should be sent to the interviewee, expressing appreciation for his or her time and cooperation. In the memo, note the date, time, location, purpose of the interview, and the main points discussed, so the interviewee has a written summary and can offer additions or corrections.

### Step 7: Evaluate the Interview

In addition to recording the facts obtained in an interview, try to identify any possible biases. For example, an interviewee who tries to protect his or her own area or function might give incomplete answers or refrain from volunteering information. Or, an interviewee with strong opinions about the current or future system might distort the facts. Some interviewees might answer questions in an attempt to be helpful even though they do not have the necessary experience to provide accurate information.

Some interviews are unsuccessful irrespective of the amount of preparation. One of the main reasons could be that the interviewer and the interviewee did not get along well. Such a situation can be caused by several factors. For example, a misunderstanding or personality conflict could affect the interview negatively, or the interviewee might be afraid that the new system will eliminate or change his or her job.

In other cases, the interviewee might give only short or incomplete responses to the open-ended questions. If so, switching to closed-ended questions, or questions with a range of replies, may elicit more favorable responses. If that still does not help, find a tactful way to conclude the meeting.

Continuing an unproductive interview is difficult. The interviewee could be more cooperative later, or the analyst might find the information required elsewhere. If failure to obtain specific information will jeopardize the success of the project, the supervisor should be informed, who can help decide what action to take. The supervisor might contact the interviewee’s supervisor, ask another systems analyst to interview the person, or find some other way to get the needed information.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.5**Gathering Requirements Using Other Techniques

In addition to interviewing, systems analysts use other requirement gathering techniques, including document review, observation, questionnaires and surveys, sampling, and research. Such techniques are used before interviewing begins to obtain a good overview and to help develop better interview questions.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.1Document Review

[**Document review**](javascript://) can help the analyst understand how the current system is supposed to work. Remember that system documentation sometimes is out of date. Forms can change or be discontinued, and documented procedures often are modified or eliminated. It is prudent to obtain copies of actual forms and operating documents currently in use, and to review blank copies of forms, as well as samples of actual completed forms. Document samples can be obtained during interviews with the people who perform that procedure. If the system uses a software package, review the documentation for that software.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.2Observation

The [**observation**](javascript://) of current operating procedures is another fact-finding technique. Seeing the system in action provides additional perspective and a better understanding of system procedures. Personal observation also allows the analyst to verify statements made in interviews and determine whether procedures really operate as they are described. Through observation, it might be discovered that neither the system documentation nor the interview statements are accurate.

Personal observation also can provide important advantages as the development process continues. For example, recommendations often are better accepted when they are based on personal observation of actual operations. Observation also can provide the knowledge needed to test or install future changes and can help build relationships with the users who will work with the new system.

Plan observations in advance by preparing a checklist of specific tasks to observe and questions to ask. Consider the following issues when preparing the list:

* Ask sufficient questions to ensure a complete understanding of the present system operation. A primary goal is to identify the methods of handling situations that are not covered by standard operating procedures. For example, what happens in a payroll system if an employee loses a time card? What is the procedure if an employee starts a shift 10 minutes late but then works 20 minutes overtime? Often, the rules for exceptions such as these are not written or formalized; therefore, try to document any procedures for handling exceptions.
* Observe all the steps in a transaction and note the documents, inputs, outputs, and processes involved.
* Examine each form, record, and report. Determine the purpose each item of information serves.
* Consider each user who works with the system and the following questions: What information does that person receive from other people? What information does this person generate? How is the information communicated? How often do interruptions occur? How much downtime occurs? How much support does the user require, and who provides it?
* Talk to the people who receive current reports to see whether the reports are complete, timely, accurate, and in a useful form. Ask whether information can be eliminated or improved and whether people would like to receive additional information.

As people are observed at work, as shown in [Figure 4-11](javascript://), consider a factor called the [**Hawthorne Effect**](javascript://). The name comes from a well-known study performed in the Hawthorne plant of the Western Electric Company in the 1920s. The purpose of the study was to determine how various changes in the work environment would affect employee productivity. The surprising result was that productivity improved during observation whether the conditions were made better or worse. Researchers concluded that productivity seemed to improve whenever the workers knew they were being observed.

**Figure 4-11**

The Hawthorne study suggested that worker productivity improves during observation. Always consider the Hawthorne Effect when observing the operation of an existing system.



Enlarge Image

Monkey Business Images/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

Although some recent studies have raised questions about the original findings, be aware that observation can and does have an effect on normal operations. With this in mind, always give advance notice to the supervisor in that area. In some situations, it might be helpful to explain the purpose of the visit to the people being observed.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

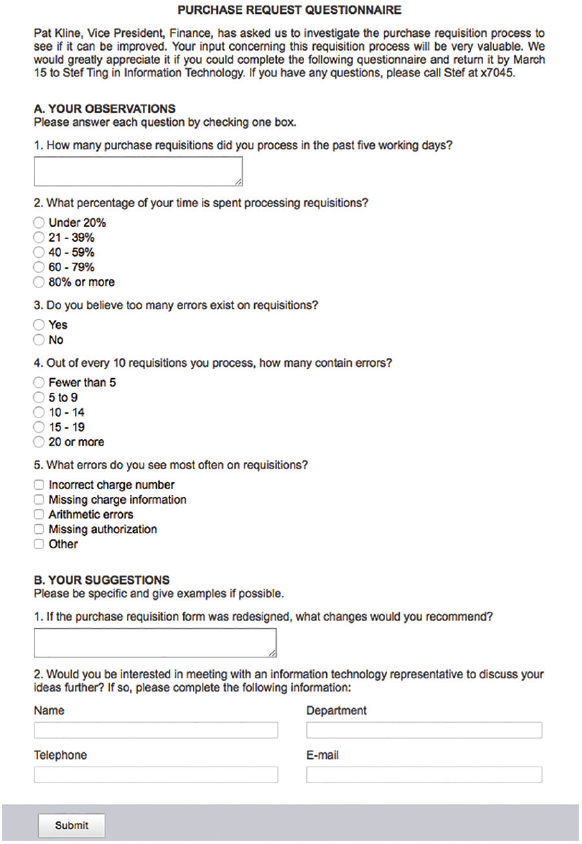
## 4.5.3Questionnaires and Surveys

In projects where it is desirable to obtain input from a large number of people, a questionnaire can be a valuable tool. A [**questionnaire**](javascript://), also called a [**survey**](javascript://), is a document containing a number of standard questions that can be sent to many individuals.

Questionnaires can be used to obtain information about a wide range of topics, including workloads, reports received, volumes of transactions handled, job duties, difficulties, and opinions of how the job could be performed better or more efficiently. [Figure 4-12](javascript://) shows a sample questionnaire that includes several different question and response formats.

**Figure 4-12**

Online version of a sample questionnaire. Does it follow the suggested guidelines?



**Source:** Created by author using Adobe Online Forms

A typical questionnaire starts with a heading, which includes a title, a brief statement of purpose, the name and telephone number of the contact person, the deadline date for completion, and how and where to return the form. The heading usually is followed by general instructions that provide clear guidance on how to answer the questions. Headings also are used to introduce each main section or portion of the survey and include instructions when the type of question or response changes. A long questionnaire might end with a conclusion that thanks the participants and reminds them how to return the form.

What about the issue of anonymity? Should people be asked to sign the questionnaire, or is it better to allow anonymous responses? The answer depends on two questions. First, does an analyst really need to know who the respondents are in order to match or correlate information? For example, it might be important to know what percentage of users need a certain software feature, but specific usernames might not be relevant. Second, does the questionnaire include any sensitive or controversial topics? Many people do not want to be identified when answering a question such as “How well has your supervisor explained the system to you?” In such cases, anonymous responses might provide better information.

When designing a questionnaire, the most important rule of all is to make sure that the questions collect the right data in a form that can be used to further the fact-finding effort. Here are some additional ideas to keep in mind when designing the questionnaire:

* Keep the questionnaire brief and user-friendly.
* Provide clear instructions that will answer all anticipated questions.
* Arrange the questions in a logical order, going from simple to more complex topics.
* Phrase questions to avoid misunderstandings; use simple terms and wording.
* Try not to lead the response or use questions that give clues to expected answers.
* Limit the use of open-ended questions that are difficult to tabulate.
* Limit the use of questions that can raise concerns about job security or other negative issues.
* Include a section at the end of the questionnaire for general comments.
* Test the questionnaire whenever possible on a small test group before finalizing it and distributing to a large group.

A questionnaire can be a traditional paper form, or it can be created in a [**fill-in form**](javascript://), and the data can be collected on the Internet or a company intranet. Before publishing the form, protect it so users can fill it in but cannot change the layout or design. Online survey websites, such as SurveyMonkey and Google Forms, can also be used to create and manage questionnaires.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.4Interviews Versus Questionnaires

When seeking input from a large group, a questionnaire is a very useful tool. On the other hand, if detailed information is required from only a few people, then each person should probably be interviewed individually. Is it better to interview or use a questionnaire? Each situation is different; consider the type of information, time constraints, and expense factors.

The interview is more familiar and personal than a questionnaire. People who are unwilling to put critical or controversial comments in writing might talk more freely in person. Moreover, during a face-to-face interview, the interviewer can react immediately to anything the interviewee says. If surprising or confusing statements are made, the topic can be pursued with additional questions. In addition, during a personal interview, the analyst can watch for clues to help determine if responses are knowledgeable and unbiased. Participation in interviews also can affect user attitudes because people who are asked for their opinions often view the project more favorably.

Interviewing, however, is a costly and time-consuming process. In addition to the meeting itself, both people must prepare, and the interviewer has to do follow-up work. When a number of interviews are planned, the total cost can be quite substantial. The personal interview usually is the most expensive fact-finding technique.

In contrast, a questionnaire gives many people the opportunity to provide input and suggestions. Questionnaire recipients can answer the questions at their convenience and do not have to set aside a block of time for an interview. If the questionnaire allows anonymous responses, people might offer more candid responses than they would in an interview.

Preparing a good questionnaire, however, like a good interview, requires skill and time. If a question is misinterpreted, its meaning cannot be clarified as easily as in a face-to-face interview. Furthermore, unless questionnaires are designed well, recipients might view them as intrusive, time-consuming, and impersonal. The analyst should select the technique that will work best in a particular situation.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.5Brainstorming

Another popular method of obtaining input is called [**brainstorming**](javascript://), which refers to a small group discussion of a specific problem, opportunity, or issue. This technique encourages new ideas, allows team participation, and enables participants to build on each other’s inputs and thoughts. Brainstorming can be structured or unstructured. In [**structured brainstorming**](javascript://), each participant speaks when it is his or her turn or passes. In [**unstructured brainstorming**](javascript://), anyone can speak at any time. At some point, the results are recorded and made part of the fact-finding documentation process.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.6Sampling

When studying an information system, examples of actual documents should be collected using a process called [**sampling**](javascript://). The samples might include records, reports, operational logs, data entry documents, complaint summaries, work requests, and various types of forms. Sampling techniques include systematic sampling, stratified sampling, and random sampling.

Suppose there is a list of 200 customers who complained about errors in their statements, and a representative sample of 20 customers will be reviewed. A [**systematic sample**](javascript://) would select every tenth customer for review. To ensure that the sample is balanced geographically, however, a [**stratified sample**](javascript://) could be used to select five customers from each of the four postal codes. Another example of stratified sampling is to select a certain percentage of transactions from each postal code, rather than a fixed number. Finally, a [**random sample**](javascript://) selects any 20 customers.

The main objective of a sample is to ensure that it represents the overall population accurately. If inventory transactions are being analyzed, for example, select a sample of transactions that is typical of actual inventory operations and does not include unusual or unrelated examples. For instance, if a company performs special processing on the last business day of the month, that day is not a good time to sample typical daily operations. To be useful, a sample must be large enough to provide a fair representation of the overall data.

Sampling should also be considered when using interviews or questionnaires. Rather than interviewing everyone or sending a questionnaire to the entire group, a sample of participants can be used. Sound sampling techniques must be used to reflect the overall population and obtain an accurate picture.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.5.7Research

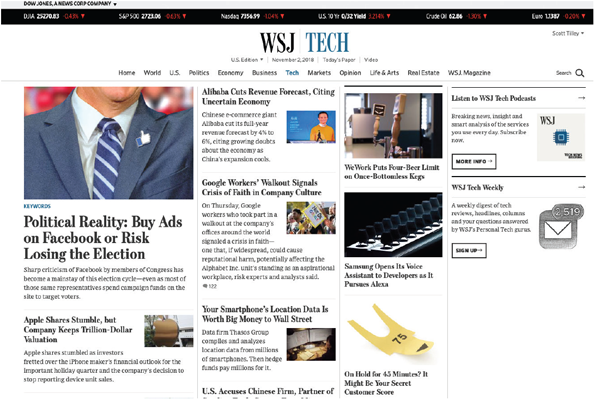
[**Research**](javascript://) is another important fact-finding technique. Research can include the Internet, IT magazines, and books to obtain background information, technical material, and news about industry trends and developments. In addition, attending professional meetings, seminars, and discussions with other IT professionals can be very helpful in problem solving.

The Internet is an extremely valuable resource. Using the Internet, the analyst can access information from federal and state governments as well as from publishers, universities, and libraries around the world. Online forums and newsgroups are good resources for exchanging information with other professionals, seeking answers to questions, and monitoring discussions that are of mutual interest.

All major hardware and software vendors maintain websites with information about products and services offered by the company. There are also websites maintained by publishers and independent firms that provide links to hundreds of hardware and software vendors. Examples of popular websites for IT professionals include Ars Technica, CNET, InfoWorld, TechCrunch, and the Wall Street Journal’s Technology pages (shown in [Figure 4-13](javascript://)).

**Figure 4-13**

The Wall Street Journal’s Technology website contains valuable information for IT professionals.



Enlarge Image

**Source:** The Wall Street Journal

Research also can involve a visit to a physical location, called a [**site visit**](javascript://), where the objective is to observe a system in use at another location. For example, if a firm’s human resources information system is the subject of study, it might be beneficial to see how another company’s system works. Site visits also are important when considering the purchase of a software package. If the software vendor suggests possible sites to visit, be aware that such sites might constitute a biased sample. A single site visit seldom provides true pictures, so try to visit more than one installation.

Before a site visit, prepare just as for an interview. Contact the appropriate manager and explain the purpose of the visit. Decide what questions will be asked and what processes will be observed. During the visit, observe how the system works and note any problems or limitations. Also learn about the support provided by the vendor, the quality of the system documentation, and so on.

**Case in Point 4.2**

### CyberStuff

* CyberStuff is a large company that sells computer hardware and software via telephone and online. CyberStuff processes several thousand transactions per week on a three-shift operation and employs 50 full-time and 125 part-time employees. Lately, the billing department has experienced an increase in the number of customer complaints about incorrect bills. You have been tasked with finding out why this is happening.

During your preliminary investigation, you discovered that some CyberStuff representatives did not follow established order entry procedures. You feel that with more information, you might find a pattern and identify a solution for the problem, but you are not sure how to proceed.

Is a questionnaire the best approach, or would interviews be better? And whether you use interviews, a questionnaire, or both techniques, should you select the participants at random, include an equal number of people from each shift, or use some other approach? How do you proceed?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.6**Gathering Requirements in Agile Projects

If agile methods are used for requirements gathering, a variation on interviews that focuses on [**features**](javascript://), [**user stories**](javascript://), [**scenarios**](javascript://), and [**storyboards**](javascript://) is used. A feature (sometimes called an [**epic**](javascript://)) is a simple, high-level statement of a requirement. A feature has a descriptive name, an estimate of its size in terms of derived requirements or user stories, and a priority. Features are typically provided by the stakeholders through initial interviews with the systems analyst.

User stories represent more fine-grained requirements. Taken together, a set of user stories forms a feature. A user story also has a descriptive name, along with a simple sentence of the form “As a [user role], I want [action] so that [goal].” *User roles*, *actions*, and *goals* are terms that represent a category of stakeholder, a particular effect, and an outcome, respectively. User stories also include an optional condition of satisfaction, which can be used as a guide to determine whether or not the requirement was satisfied by the product. A user story is meant to be succinct and is often drawn on a  index card (or the software equivalent).

A scenario is a real-world example of how users will interact with the system. A scenario describes a particular set of steps taken or events that will occur while the system is used for a specific function. It is used to refine the system requirements to better reflect actual usage of the system.

A storyboard is a simple graphic organizer that helps systems analysts visualize the status of a project. A storyboard can be as rudimentary as a wall with sticky notes. There are also many software tools that enhance storyboards while still maintaining their essential feel.

Because agile methods are iterative in nature, requirements are gathered and successively refined. They begin as features, which are then split into smaller user stories, which in turn are refined into scenarios. Agile methods are particularly well suited to projects where the requirements are expected to change frequently.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.7**Representing Requirements

Once requirements have been gathered, they must be recorded. Keeping accurate records of interviews, facts, ideas, and observations is essential to successful systems development. As information is gathered, the importance of a single item can be overlooked or complex system details can be forgotten. The ability to manage information is the mark of a successful systems analyst and an important skill for all IT professionals. The basic rule is to write it down. Analysts should document their work according to the following principles:

* Record information as soon as it is obtained.
* Use the simplest recording method possible.
* Record findings in such a way that someone else can understand them.
* Organize documentation so related material is located easily.

There are several techniques used to do this, ranging from an unordered collection of sentences to a structured database of formal models. Whatever representation is chosen, properly managing the requirements over the lifetime of a project is a key to its success.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.7.1Natural Language

The vast majority of requirements are represented using unstructured natural language, that is, plain English. Examples of these requirements were shown in [Section 4.1.1](javascript://). This representation is easy to create but is prone to problems such as imprecision and a lack of shared understanding.

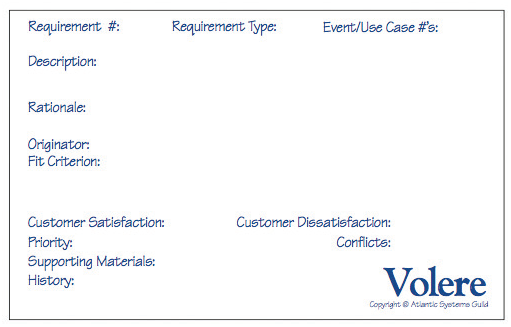
Requirements represented as unstructured natural language can be stored in a simple file, in a database that may facilitate searching its contents, or in an Excel spreadsheet. The latter choice is a popular one, because most systems analysts are familiar with the tool, and they are likely to have access to it already.

An improvement upon unstructured natural language is structured natural language. This representation tags part of the requirement, rather like an XML document. This facilitates automated processing of the requirement statements, but it’s not very user-friendly.

Requirements represented as structured natural language can be stored on a simple index card. The Volere shell from the Atlantic Systems Guild, shown in [Figure 4-14](javascript://), is a mature method of representing single requirements on a  card. There are several tools that mimic these features, providing automation while maintaining the simple representational format.

**Figure 4-14**

The Volere shell.



**Source:** Atlantic Systems Guild

There are also formal techniques, based on mathematics, that can be used to represent complex requirements. Languages such as VDM and Z are used for this purpose. They have the advantage of mathematical consistency, but a major disadvantage of this technique is that many systems analysts are unfamiliar with the language’s constructs, which can affect the requirements’ understandability.

Irrespective of the natural language technique chosen, individual requirements must be collected into a requirements document. This document is the official statement of what is required of the system by the users. The Volere template is a good template for a comprehensive requirements document. It is populated by atomic requirements from the Volere shells.

**Case in Point 4.3**

### Digital Pen Transcription

* You are the lead systems analyst on a large project. A big part of your responsibilities is to handle the system requirements. Extensive interviews have taken place, and you have written down your notes using a digital pen system. This system recognizes letters and words while you write on paper and stores them in digital format on your computer. In this way, the requirements are written in unstructured natural language but stored digitally, which means they can be searched and processed. Do you think it would be worthwhile to edit the transcription and insert tags to delineate key terms in the requirements, so that a CASE tool could better analyze them? There’s a lot of effort required to do the manual tagging, but the rewards could be better requirements. How would you perform the tradeoff analysis?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.7.2Diagrams

Many people are more visual than textual. For them, diagrams are an excellent choice to represent system requirements. Diagrams help users, managers, and IT professionals understand the design of a system. Diagramming involves graphical methods and nontechnical language that represent the system at various stages of development. During requirements engineering, the analyst can use various tools to describe business processes, requirements, and user interaction with the system.

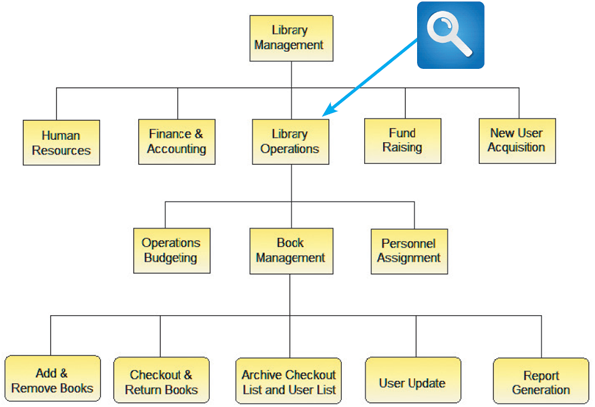
Systems analysts use diagramming and fact-finding interactively—first they build fact-finding results into diagrams, then they study the diagrams to determine whether additional fact-finding is needed. To help them understand system requirements, analysts use functional decomposition diagrams, business process diagrams, and data flow diagrams. Any of these diagrams can be created with CASE tools or stand-alone drawing tools if desired.

### Functional Decomposition Diagrams

A [**functional decomposition diagram (FDD)**](javascript://) is a top-down representation of a function or process. Using an FDD, an analyst can show business functions and break them down into lower-level functions and processes. Creating an FDD is similar to drawing an organization chart: Start at the top and work downward. [Figure 4-15](javascript://) shows an FDD of a library system drawn with the Visible Analyst CASE tool. FDDs can be used at several stages of systems development. During requirements engineering, analysts use FDDs to model business functions and show how they are organized into lower-level processes. These processes translate into program modules during application development.

**Figure 4-15**

This Visible Analyst FDD shows a library system with five top-level functions. The Library Operations function includes two additional levels of processes and sub-processes.



Enlarge Image

**Source:** Screenshot used with permission from Visible Systems Corporation.

### Business Process Diagrams

As described in [Chapter 1](javascript://), a **business process model (BPM)** represents one or more business processes, such as handling an airline reservation, filling a product order, or updating a customer account. During requirements engineering, analysts often create diagrams that use a standard syntax called **business process modeling notation (BPMN)**. BPMN includes various shapes and symbols to represent events, processes, and workflows.

When creating a business process diagram using a CASE tool such as Visible Analyst, the diagram automatically becomes part of the overall model. In the example shown in [Figure 4-16](javascript://), using BPMN terminology, the overall diagram is called a [**pool**](javascript://) and the designated customer areas are called [**swim lanes**](javascript://). Integrating BPM into the CASE development process leads to faster results, fewer errors, and reduced cost.

**Figure 4-16**

Using the Visible Analyst CASE tool, an analyst can create a business process diagram. The overall diagram is called a pool, and the two separate customer areas are called swim lanes.

Enlarge Image

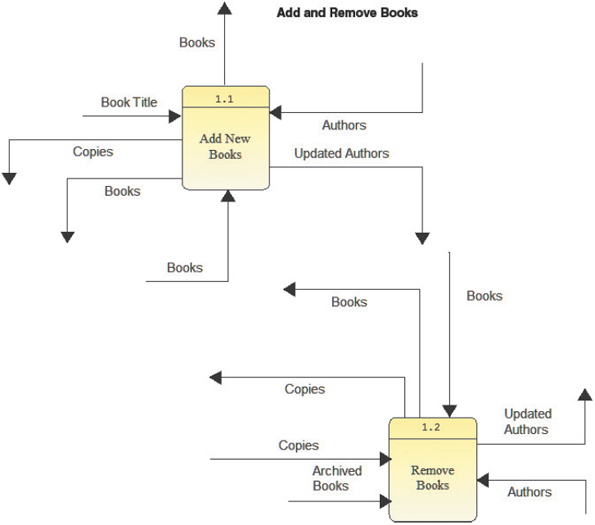
**Source:** Screenshot used with permission from Visible Systems Corporation.

### Data Flow Diagrams

Working from an FDD, analysts can create **data flow diagrams (DFDs)** to show how the system stores, processes, and transforms data. The DFD in [Figure 4-17](javascript://) describes adding and removing books, which is a function shown in the Library Management diagram in [Figure 4-15](javascript://). Note that the two boxes in the DFD represent processes, each with various inputs and outputs. Additional levels of information and detail are depicted in other, related DFDs. Data and process modeling is described in detail in [Chapter 5](javascript://).

**Figure 4-17**

This Visible Analyst DFD shows how books are added and removed in a library system.



Enlarge Image

**Source:** Screenshot used with permission from Visible Systems Corporation

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 4.7.3Models

Models provide a more formal representation of system requirements. They are often depicted as graphical in nature, so they share some of the characteristics of the techniques described in [Section 4.7.2](javascript://). But models have an additional feature: The underlying language has semantics, which means the diagram has a significance that can be automatically analyzed by a CASE tool.

The [**Unified Modeling Language (UML)**](javascript://) is perhaps the most widely used modeling technique for visualizing and documenting software systems design. UML uses object-oriented design concepts, but it is independent of any specific programming language and can be used to describe business processes and requirements generally. **[SysML](javascript://)** is a dialect of UML and has become the standard for Model-Based Systems Engineering (MBSE) applications.

UML provides various graphical tools, such as use case diagrams and sequence diagrams. During requirements engineering, a systems analyst can utilize the UML to represent the information system from a user’s viewpoint. Use case diagrams, sequence diagrams, and other UML concepts are discussed in more detail in [Chapter 6](javascript://), along with other object-oriented analysis concepts. A brief description of each technique follows.

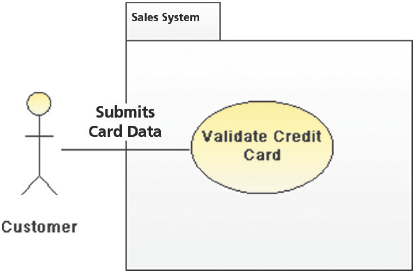
### Use Case Diagrams

During requirements engineering, systems analysts and users work together to document requirements and model system functions. A [**use case diagram**](javascript://) visually represents the interaction between users and the information system. In a use case diagram, the user becomes an [**actor**](javascript://), with a specific role that describes how he or she interacts with the system. Systems analysts can draw use case diagrams freehand or use CASE tools that integrate the use cases into the overall systems design.

[Figure 4-18](javascript://) shows a simple use case diagram for a sales system where the actor is a customer and the use case involves a credit card validation that is performed by the system. Because use cases depict the system through the eyes of a user, common business language can be used to describe the transactions. For example, [Figure 4-19](javascript://) shows a table that documents the credit card validation use case, and [Figure 4-20](javascript://) shows a student records system, with several use cases and actors.

**Figure 4-18**

This Visible Analyst use case diagram shows a sales system, where the actor is a customer and the use case is a credit card validation.



**Source:** Screenshot used with permission from Visible Systems Corporation.

**Figure 4-19**

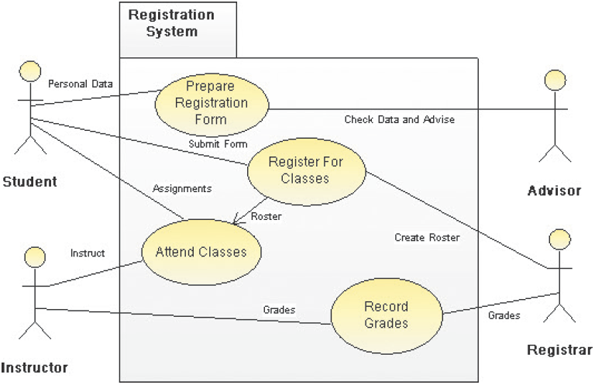
This table documents the credit card validation use case shown in [Figure 4-18](javascript://).

|  |  |
| --- | --- |
| **Name of Use Case:** | Credit card validation process |
| **Actor:** | Customer |
| **Description:** | Describes the credit card validation process |
| **Successful Completion:** | 1. Customer clicks the input selector and enters credit card number and expiration date 2. System verifies card 3. System sends authorization message |
| **Alternative:** | 1. Customer clicks the input selector and enters credit card number and expiration date 2. System rejects card 3. System sends rejection message |
| **Precondition:** | Customer has selected at least one item and has proceeded to checkout area |
| **Postcondition:** | Credit card information has been validated  Customer can continue with order |
| **Assumptions:** | None |

Enlarge Table

**Figure 4-20**

This Visible Analyst use case diagram shows a student records system.



Enlarge Image

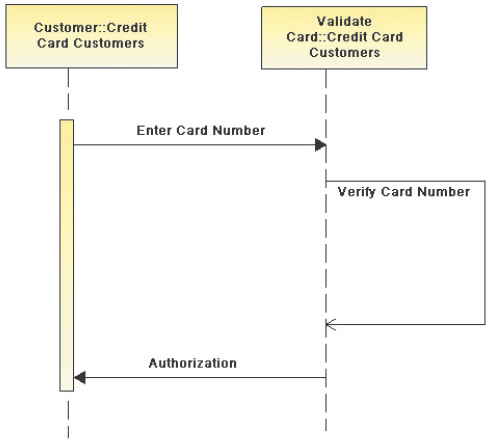
**Source:** Screenshot used with permission from Visible Systems Corporation.

### Sequence Diagram

A [**sequence diagram**](javascript://) shows the timing of interactions between objects as they occur. A systems analyst might use a sequence diagram to show all possible outcomes or focus on a single scenario. [Figure 4-21](javascript://) shows a simple sequence diagram of a successful credit card validation. The interaction proceeds from top to bottom along a vertical timeline, while the horizontal arrows represent messages from one object to another.

**Figure 4-21**

This Visible Analyst sequence diagram shows a credit card validation process.



**Source:** Screenshot used with permission from Visible Systems Corporation.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.8**Validating and Verifying Requirements

Requirements validation and verification (V&V) is concerned with demonstrating that the requirements define the system that the customer really wants. Since requirements error costs are high, V&V is very important; it is many times more expensive to fix a system later in the SDLC than it is to fix it during requirements engineering.

Requirements V&V focuses on answering two important questions:

* Validation: Are the correct requirements stated?
* Verification: Are the requirements stated correctly?

To answer these questions, the following requirements attributes should be checked:

* Validity: Does the system provide the functions that best support the customer’s needs?
* Consistency: Are there conflicting requirements?
* Completeness: Are all functions required by the customer included?
* Realism: Can the requirements be implemented given available budget and technology?
* Verifiability: Can the requirements be checked?
* Comprehensibility: Is the requirement properly understood?
* Traceability: Is the origin of the requirement clearly stated?
* Adaptability: Can the requirement be changed without a large impact on other requirements?

To check these, the following techniques can be used:

* Requirements reviews: Systematic manual analysis of the requirements.
* Prototyping: Using an executable model of the system to check the requirements.
* Test-case generation: Developing tests for requirements to check testability.
* Automated consistency analysis: Checking the consistency of a structured or formal requirements descriptions.

Consider requirements reviews. Regular reviews can be held while the requirements are initially being formulated. Ideally, the systems analyst and the customer (and perhaps other key stakeholders) should be involved in the reviews. The reviews can be formal (with complete documentation) or informal. The key is that good communications between analysts, customers, and others can resolve problems at an early stage, which is better for everyone.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

# 4.9Tools

All requirements engineering activities can be helped through the judicious use of tools. For example, many software programs are available to help record and document information elicited during the requirements gathering process. This type of [**productivity software**](javascript://) includes automation, word processing, spreadsheet, database management, presentation graphics, and collaboration software programs. Although Microsoft Office is the best-known set of productivity software programs, other vendors offer products in each of these categories.

A [**personal information manager (PIM)**](javascript://), such as Microsoft Outlook, includes a personal calendar, a to-do list with priorities and the capability to check off completed items, and powerful contact management features. Outlook can manage email and appointments and supports collaboration and team projects.

Although a PIM such as Microsoft Outlook can handle day-to-day activities, tasks, and schedules, it is not the best way to capture and organize large amounts of information. Instead, analysts use information management software such as Microsoft OneNote, which is a powerful, flexible tool that can handle many different types of input, including text, handwritten notes, images, audio and video recording, web links, and much more. OneNote is included in several versions of the Office suite.

[Figure 4-22](javascript://) shows another popular PIM called Evernote. It is available for free on most computing platforms, including smartphones and on the web. There are also premium versions available on a monthly subscription model. Evernote does a great job of handling all sorts of multimedia content, adding free-form notes, and providing templates for organizing projects. It also syncs files across all devices.

**Figure 4-22**

Evernote offers a free version of its popular information management software for most computing platforms, including smartphones and on the web.

Enlarge Image

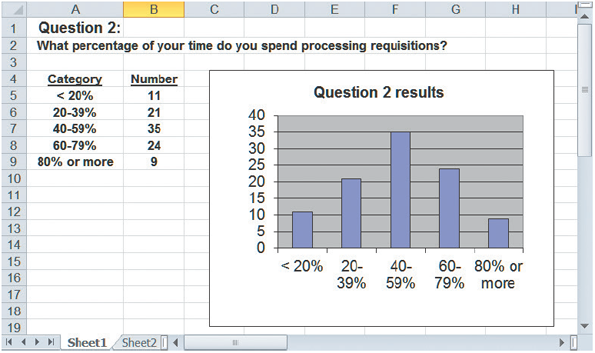
**Source:** Evernote

Using word processing software such as Microsoft Word, the analyst can create reports, summaries, tables, and forms. In addition to standard document preparation, the program can help organize a presentation with templates, bookmarks, annotations, revision control, and an index. Fill-in forms can also be created to conduct surveys and questionnaires, as described earlier in this chapter.

Spreadsheet software, such as Microsoft Excel, can help track and manage numeric data or financial information. In fact, Excel is one of the most popular ways of representing requirements in an informal manner. Graphs and charts can also be generated that display the data and show possible patterns. The statistical functions in a spreadsheet can be used to tabulate and analyze questionnaire data. A graphical format often is used in quality control analysis because it highlights problems and their possible causes, and it is effective when presenting results to management. A common tool for showing the distribution of questionnaire or sampling results is a vertical bar chart called a [**histogram**](javascript://). Most spreadsheet programs can create histograms and other charts that can display the data collected. [Figure 4-23](javascript://) displays a typical histogram that might have resulted from the questionnaire shown in [Figure 4-12](javascript://).

**Figure 4-23**

This histogram displays the results from Question 2 in the questionnaire shown in [Figure 4-12](javascript://).



Enlarge Image

Database management software allows the analyst to document and organize fact-finding results such as events, observations, and data samples. A database program such as Microsoft Access can be used to manage the details of a complex project, create queries to retrieve specific information, and generate custom reports.

Presentation graphics software, such as Microsoft PowerPoint, is a powerful tool for organizing and developing formal presentations. Presentation graphics programs enable the creation of organization charts that can be used in a preliminary investigation and later during requirements engineering. These high-quality charts also can be included in written reports and management presentations.

Collaboration software is the latest weapon in the struggle to boost productivity. People work in teams and use web-based software such as Google Docs and Microsoft Office 365 to access data and share files. Google and others are betting that cloud computing will create a virtual workplace, where people will be able to interact in real time, with all the benefits of a traditional face-to-face workplace but none of the limitations.

When it comes to creating diagrams that represent requirements, Microsoft Visio is a popular graphic modeling tool that can produce a wide range of charts and diagrams. Visio includes a library of templates, stencils, and shapes. An analyst can use Visio to create many types of visual models, including business processes, flowcharts, network diagrams, organization charts, and many more. For example, in [Figure 4-24](javascript://), the analyst used drag-and-drop shapes to represent a business process.

**Figure 4-24**

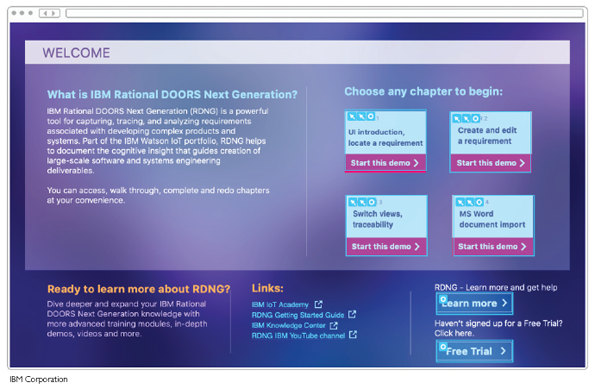
This Visio drawing uses drag-and-drop shapes to represent a business process.

Enlarge Image

For more formal models of requirements, special-purpose tools such as IBM Rational DOORS are used. These tools facilitate the use of UML (and SysML) to model system requirements in a way that enables desirable characteristics, such as [**traceability**](javascript://), where the origin of a requirement is connected back to the requirement itself, which in turn is linked forward to design artifacts, code fragments, and even test cases in the SDLC. [Figure 4-25](javascript://) illustrates some of the capabilities of DOORS Next Generation.

**Figure 4-25**

IBM DOORS is a tool for capturing, analyzing, and tracing system requirements.



Enlarge Image

IBM Corporation

### A Question of Ethics

* [iStock.com](https://istock.com/" \t "_blank)/faberfoto\_itYour supervisor manages the corporate office where you work as a systems analyst. Several weeks ago, after hearing rumors of employee dissatisfaction, he asked you to create a survey for all IT employees. After the responses were returned and tabulated, he was disappointed to learn that many employees assigned low ratings to morale and management policies.

This morning he called you into his office and asked whether you could identify the departments that submitted the lowest ratings. No names were used on the individual survey forms. However, with a little analysis, you probably could identify the departments because several questions were department related.

Now you are not sure how to respond. The expectation was that the survey would be anonymous. Even though no individuals would be identified, would it be ethical to reveal which departments sent in the low ratings? Would your supervisor’s motives for wanting this information matter?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**4.10**Summary

The systems analysis phase includes three activities: requirements engineering, data and process modeling, and consideration of development strategies. The main objective is to understand the proposed project, ensure that it will support business requirements, and build a solid foundation for the systems design phase. Requirements engineering itself is composed of three main parts:

* (1)

gathering requirements,

* (2)

representing requirements, and

* (3)

validating and verifying requirements.

During requirements engineering, the business-related requirements for the new information system are identified. Scalability is considered to ensure that the system can support future growth and expansion. Security is an essential requirement of all modern connected systems. The TCO is also estimated to identify all costs, including indirect costs.

Popular team-based approaches include JAD, RAD, and agile methods. JAD is a popular, team-based approach to fact-finding and requirements engineering. JAD involves an interactive group of users, managers, and IT professionals who participate in requirements engineering and develop a greater commitment to the project and to their common goals.

RAD is a team-based technique that speeds up information systems development and produces a functioning information system. RAD is a complete methodology, with a four-phase life cycle that parallels the traditional SDLC phases.

Agile methods attempt to develop a system incrementally by building a series of prototypes and constantly adjusting them to user requirements. Tools are often avoided and replaced with simpler aids, such as whiteboards and sticky notes to facilitate communication.

The requirements gathering process includes interviewing, document review, observation, questionnaires, sampling, and research. Successful interviewing requires good planning and strong interpersonal and communication skills. The systems analyst must decide on the people to interview; set interview objectives; and prepare for, conduct, and analyze interviews. The analyst also might find it helpful to use one or more software tools during fact-finding.

Systems analysts use various tools and techniques to represent system requirements. Natural language, structured or unstructured, is still the default. Requirements can be stored as simple text in a plain file or in an Excel spreadsheet. They can also be stored online for automated search and analysis.

Diagrams are another way of representing requirements. They are suitable for analysts who are more visually oriented. They can also capture complementary aspects of the system requirements. Sample diagram types include FDDs, business process diagrams, and DFDs.

Models provide a more formal representation of system requirements. The UML is a widely used method of visualizing and documenting software design through the eyes of the business user. UML tools include use case diagrams and sequence diagrams to represent actors, their roles, and the sequence of transactions that occurs.

Systems analysts should carefully record and document factual information as it is collected, and various software tools can help an analyst visualize and describe an information system.

Requirements V&V is concerned with demonstrating that the requirements define the system that the customer really wants. Validation asks if the correct requirements are stated, while verification asks if the requirements are stated correctly.

All requirements engineering activities can be helped through the judicious use of tools. They provide automated support for requirements attributes such as traceability. For very large systems with thousands of requirements to manage, CASE tool assistance is necessary.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

# Chapter Review

## **Key Terms**

* [**actor**](javascript://)
* **agile methods**
* [**brainstorming**](javascript://)
* **business process model (BPM)**
* **business process modeling notation (BPMN)**
* [**closed-ended questions**](javascript://)
* [**construction phase**](javascript://)
* [**cutover phase**](javascript://)
* **data flow diagram (DFD)**
* [**document review**](javascript://)
* [**engaged listening**](javascript://)
* [**epic**](javascript://)
* [**fact-finding**](javascript://)
* [**features**](javascript://)
* [**fill-in form**](javascript://)
* [**functional decomposition diagram (FDD)**](javascript://)
* [**functional requirement**](javascript://)
* [**Hawthorne Effect**](javascript://)
* [**histogram**](javascript://)
* [**Informal structures**](javascript://)
* [**interview**](javascript://)
* [**joint application development (JAD)**](javascript://)
* [**leading questions**](javascript://)
* [**non-functional requirement**](javascript://)
* [**observation**](javascript://)
* [**open-ended questions**](javascript://)
* [**personal information manager (PIM)**](javascript://)
* [**pool**](javascript://)
* [**productivity software**](javascript://)
* [**quality attributes**](javascript://)
* [**questionnaire**](javascript://)
* [**random sample**](javascript://)
* [**range-of-response questions**](javascript://)
* [**rapid application development (RAD)**](javascript://)
* [**requirements definitions**](javascript://)
* [**requirements elicitation**](javascript://)
* **requirements engineering**
* [**requirements planning phase**](javascript://)
* [**requirements specifications**](javascript://)
* [**research**](javascript://)
* [**sampling**](javascript://)
* [**Scalability**](javascript://)
* [**scenarios**](javascript://)
* [**Scrum**](javascript://)
* [**sequence diagram**](javascript://)
* [**site visit**](javascript://)
* [**stratified sample**](javascript://)
* [**storyboards**](javascript://)
* [**structured brainstorming**](javascript://)
* [**survey**](javascript://)
* [**swim lanes**](javascript://)
* [**SysML**](javascript://)
* [**system requirement**](javascript://)
* [**systematic sample**](javascript://)
* **total cost of ownership (TCO)**
* [**traceability**](javascript://)
* [**Unified Modeling Language (UML)**](javascript://)
* [**unstructured brainstorming**](javascript://)
* [**use case diagram**](javascript://)
* [**user design phase**](javascript://)
* [**user stories**](javascript://)

Change font size

[**help**](javascript://)