Chapter 3: *Players in the Systems Game*

All the world’s a stage,
And all the men and women merely players:
They have their exits and their entrances;
And one man in his time plays many parts.

— Shakespeare
*As You Like It*, II, vii
In this chapter, you will learn:

1. The categories of people with whom you will interact in a project;
2. The three main categories of users, by job category;
3. User reactions during a systems development project;
4. The difference between novice and expert users;
5. The role of management in a systems development project;
6. The role of the systems analyst in a systems development project; and
7. Other roles in a systems development project.

As a systems analyst, you will work on systems development projects with a variety of other people. The cast of characters will change from project to project; the personalities will differ dramatically; and the number of people that you interact with will range from as few as one to as many as several dozen. However, the roles are fairly consistent, and you will see them over and over again.

To be a successful systems analyst requires more than an understanding of the technology of computers. Among other things, it requires interpersonal skills: you will be spending a great deal of your time working with other people, many of whom speak a very different “language” than you do, and many of whom will find your language of computer technology alien and frightening. Thus, it is important to know what expectations such people will have of you and what expectations you should have of them.

This chapter concentrates on the characteristics of the following major categories of “players” that you are likely to encounter in a typical systems development project:

* Users
* Management
* Auditors, quality assurance people, and “standards bearers”
* Systems analysts
* Systems designers
* Programmers
* Web designers
* Operations personnel

Each of these categories is described next.
3.1 Users

The first, and by far the most important, player in the systems game is someone known to systems analysts as a user. The user is the person (or group of people) for whom the system is being built. He or she is the person whom you will interview, often in great detail, to learn what features the new system must have to be successful. It should be emphasized that most users don’t refer to themselves as “users”; after all, the word is often used in a different context to describe people addicted to drugs. In some organizations, the data processing organization avoids the problem by using the term customer or owner to identify the user. The user is the “owner” in the sense that he or she receives, or inherits — and thus owns — the system when it is finally built. And the user is the “customer” in at least two important respects: (1) as in so many other professions, “the customer is always right,” regardless of how demanding, unpleasant, or irrational he or she may seem; and (2) the customer is ultimately the person paying for the system and usually has the right or ability to refuse to pay if he or she is unhappy with the product received.

In most cases, it is fairly easy to identify the user (or users): the user is typically the person who makes a formal request for a system. In a small organization, this is usually a very informal process; it may consist of the user picking up the phone and calling the Official Systems Analyst to say, “Hey, Joan! I need a new system to keep track of our new Widget marketing campaign!” In a large organization, the initiation of a systems development project is usually much more formalized. The “request for system survey and study,” as it is sometimes called, usually goes through several levels of approval before the systems analyst gets involved. More on this in Chapter 5.

There are a number of situations where the identity of the real user is not known, or where the systems analyst has little or no opportunity to interact directly with the user. One common example is that of a system being developed by a consulting firm or software company: the interaction between the client organization and the consulting firm may take place between contract officers or other administrative agencies, sometimes with explicit provisos that the systems analyst may not talk directly to the user. Another common example is the development of a software product, where the features and functionality are defined by the marketing department, even though they will not be the people who actually use the system. Even if the system is being developed entirely within a single organization, the “real” user may nominate a spokesperson to work with the systems analyst because he or she is too busy with other work. [1] Obviously, in situations like this there is a distinct possibility of miscommunication: whatever it is that the real user wants the system to do may not be communicated properly to the systems analyst, and whatever it is that the systems analyst thinks he or she is creating for the user may not be properly communicated — not until the entire system has been built, at which point it may be too late! There are two conclusions we can draw from this:

* Wherever possible, the systems analyst should try to establish direct contact with the user. Even if other people are involved as intermediaries (e.g., to deal with contract issues or administrative details), it is important to have regular, face-to-face meetings with the person who will ultimately inherit the system. Indeed, it is usually even better if the user is a full-time member of the project team. In many organizations, the user is the project manager; some even argue that the user should even do the project.

* If it is not possible to communicate directly with the user, then the documentation produced by the systems analyst becomes even more crucial. Part II of this book is devoted to modeling tools that can be used to describe the behavior of a system in a rigorous, formal way; it is essential to use tools of this kind to avoid costly misunderstandings.
3.1.1 The heterogeneity of users

One of the mistakes frequently made by people in the computer field — especially by computer programmers, and sometimes by systems analysts, too — is to assume that all users are the same. “User” as a singular noun implies that the systems analyst will only have to interact with one person; even when it is obvious that more than one user is involved, there is a tendency to think of them as a formless, shapeless, homogeneous group of humans. To say that one user is different from another is, of course, a trite statement: yes, they all have different personalities, different backgrounds, different interests, and so on. But there are some important differences that you must keep in mind in your work as a systems analyst. Here are two ways of categorizing users:

* Job category, or level of supervision.
* Level of experience with data processing.

3.1.2 Categorizing users by job category

On a typical systems analysis project, you will spend a considerable amount of time interviewing users to determine system requirements. But which users? At which level? Naturally, this depends on the project and on the politics within your organization — but you can usually count on interacting with three main job categories: operational users, supervisory users, and executive users. Operational users are the clerical, operational, and administrative people most likely to have the most day-to-day contact with the new system (unless you are building a decision-support system, in which case you may have little or no contact with this group). Thus, in a typical large organization, you may find yourself interviewing secretaries, insurance agents, bookkeepers, shipping clerks, order entry personnel, and “assistants” of all sizes, shapes and colors. For a real-time system, you may be talking with operational users whose titles are engineer, physicist, factory worker, pilot, telephone operator, and so on. You should keep three things in mind when you work with operational-level users:

* Operational users are very much concerned with the functions that the system will perform — but they are likely to be even more concerned with the human interface issues. Examples are: What kind of keyboard will I be using to communicate with the system; is it like the typewriter keyboard I’ve been using for so many years? What kind of on-line display screen will the system have; will there be a lot of glare, and will the characters be easy to read? How will the system tell me if I’ve made a mistake; will I have to type everything all over again? What if I want to “undo” something that I typed a few minutes ago? When the system produces a report for me, where will the information be on the page; can I get the date and time printed on the top of every page? And so on. These are issues that the operational-level user’s supervisor may or may not be aware of or interested in; but, as you can imagine, they are crucial to the success of the system and they must be addressed. This means that, as a systems analyst, you must either be allowed to communicate directly with the operational user, or (far less preferable) you must be very sure that the person who speaks on behalf of the operational user is knowledgeable about these issues. These issues are discussed in detail as part of the user implementation model in Chapter 21.
* Operational users tend to have a “local” view of the system; they tend to be knowledgeable about the specific job that they do and the people with whom they have immediate contact (customers, supervisors, colleagues, etc.). But they often are unfamiliar with the “big picture”; that is, they may have trouble describing how their activity fits into the overall organization or what the overall organization’s charter really is. This is rarely because of stupidity, but it may reflect a lack of interest on their part. Or it may reflect the fact that the supervisory user has not told them anything about the “big picture” and prefers that they not know anything about it. A consequence of this situation is that the systems analyst must be able to develop system models that permit both local views (i.e., descriptions of a small, detailed part of the system, independently of other parts) and global views (i.e., high-level overviews of the entire system that avoid the detail).

* Operational users tend to think of systems in very physical terms, that is, in terms of the implementation technology currently used to implement the system or in terms of technology that they imagine could be used. Abstract discussions about “functions” and “data elements” may be difficult; hence, the systems analyst may find it necessary to talk with the user exclusively in familiar terms. Then, as a separate activity, the analyst can translate this physical description into an “essential model” — a model of what the system must do, regardless of the technology used to implement it. This is discussed further in Chapter 17.

Supervisory users are, as the term implies, employed in a supervisory capacity: they usually manage a group of operational users and are responsible for their performance (obviously, one can imagine more than one level of supervisory user in a large organization). They may have the title of supervisor, but it might also be shift leader, foreman, office manager, branch office executive, head engineer, or a variety of other titles. The significant things to remember about supervisory users are these:

* Many of them are former operational users who have been promoted to their current position. Thus, they are generally familiar with the work done by their operational subordinates, and they can usually be expected to sympathize with their needs, concerns, and perspectives. However, this is not always true. Because the marketplace and economy and technology have changed so much, the operational job today may be very different from what it was 20 years ago.

* One reason that the supervisory user may be perceived as out of touch with the operational user is that he or she is often measured and motivated by performance against a budget. Hence the supervisory user is often most interested in a new information system because of the possibility of increasing the volume of work done, reducing the cost of processing transactions, and reducing errors in the work. And it may occur to the supervisory user that a new system will provide an opportunity to monitor the performance (and even the minute-by-minute activity) of each individual operational user. Depending on how this is implemented, the operational users may or may not have the same perspective as the supervisory user.

* Because of this emphasis on operational efficiency, it is usually the supervisory user who thinks of a new system as a way of reducing the number of operational users (by layoffs or attrition) or avoiding further increases in their numbers as the volume of work increases. This is neither good nor bad, but it is often the focal point of heated political battles, in which the systems analyst is often caught in the middle. [5]
* For the same reasons, the supervisory user will often act as a middleman between the systems analyst and the operational user, arguing that the operational users are too busy to waste their time talking to the analyst. “After all,” the supervisory user will argue, “it is precisely because we are so busy that we need a new computer system!” This is a very dangerous position to find yourself in; after all, it is the operational user who will be most concerned about the human interface of the system, and it is unlikely that the supervisory user will be able to fully empathize with those needs.
* The supervisory user often thinks in the same physical terms as the operational user, and this perspective is often just as local as that of the operational user. Of course, one would expect that a management-level person would have a somewhat more global view; as a corollary, it may turn out that the supervisory user no longer remembers some of the detailed business policy carried out by the operational users.
* Finally, it is the supervisory user with whom you will have your primary day-to-day contact. He or she is the one who will typically define the requirements and detailed business policy that your system must implement. He or she may be a passive member of the team (in the sense that he or she participates only when interviewed), a full-time member of the team, or even, as mentioned earlier, the project manager.

Executive-level users are generally not directly involved in a systems development project, unless the project is so large and so important that it has a major impact on the organization. For the normal project, though, the executive user is usually two or three levels above the action associated with the project. To the extent that you become involved with them, you will probably discover the following things about executive users:

* They may provide the initiative for the project, but are more likely to serve as the funding authority for project requests that originate at lower levels in the organization.
* They are usually not former operational users or, if they were, it was so long ago that whatever experience they had is obsolete. Thus, they are in no position to help define the requirements of the system for the people who will actually be using it on a day-to-day basis. An exception to this is the decision-support system discussed in Chapter 2; such a system would more commonly be used by the supervisory and executive users.
* Executive users are typically more concerned with strategic issues and long-term profit/loss. Hence, they are typically less concerned with such operational issues as reduced transaction costs or saving three clerical workers as they are with what Paul Strassman calls the “information payoff” in (Strassman, 1985) — that is, the new markets, new products, or new competitive advantage that they will gain from the new system. And they are also the ones most likely to be concerned about the “window of opportunity” for a new system, which often determines the deadline for the project before anyone has even determined the requirements of the system to be built.
* Executive-level users generally are more interested in a global view of the entire system; as a result, they are generally not interested in the details. As mentioned earlier, this means that we must use system modeling tools that allow us to provide an overview of the system to the executive users (and to anyone else who needs it) and detailed portions of the system to the operational users who are the “local experts.”
Similarly, executive-level users are generally able to work with abstract models of a system; indeed, they are already accustomed to working with such abstract models as financial models, marketing models, organizational models, and engineering models (of new products, factories, offices, etc.). Indeed, they will not be at all interested in “physical” models of the system and will wonder why you are bothering to show such things to them.

To summarize, then, you can expect to interact with three different types, or levels, of users, as shown in Figure 3.1. Keep in mind that they have different perspectives, different interests and priorities, and often different backgrounds. These three types of users can be characterized as shown in Table 3.1. From the previous discussion, you should be able to understand that there may be situations where user is not pleased by the prospect of a new system; indeed, sometimes they will actively oppose it. This is most often the case with the operational users (since they are the ones who will have to use it), but the resistance may also come from the supervisory user (since he or she may feel that it will have a negative impact on the efficiency or profitability of the area he or she is responsible for), or even the executive user. As Marjorie Leeson points out in (Leeson, 1981),

The analyst who understands basic motivation, why people resist change, and how they resist change, may be able to overcome some of the resistance. Most management books make reference to psychologist A.H. Maslows’ hierarchy of needs. The five categories, from lowest priority to highest, are:

<table>
<thead>
<tr>
<th>NEED</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Physiological</td>
<td>Food, clothing, and shelter</td>
</tr>
<tr>
<td>2. Safety and security</td>
<td>Protection against danger and loss of job</td>
</tr>
<tr>
<td>3. Social</td>
<td>Being able to identify with individuals and groups</td>
</tr>
<tr>
<td>4. Egotistic</td>
<td>Recognition, status, and importance</td>
</tr>
<tr>
<td>5. Self-fulfillment</td>
<td>Realizing one’s fullest potential in creativity and self-development</td>
</tr>
</tbody>
</table>

Thus, if you find some of the users resisting the idea of a new system, you should think about the possibility of one or more of these needs not being met. It is rare, of course, that a user would worry about the physiological level of need, but it is not at all surprising to find that a user is worried about the loss of his or her job. And it is also common for users (especially the operational users) to worry that a new system will lead to their not being able to identify with their familiar social groups; they fear that they will be confined to a CRT terminal all day and spend all of their time interacting with a computer rather than other humans. The operational user who has become an expert in performing an information-processing task on a manual basis may also feel that a new system will leave his or her “egotistic” needs unfulfilled; and the user who feels that the system will take away the creative aspects of his or her current work may also resist.
Figure 3.1: The three types of user

OPERATIONAL USER
Usually has a local view
Carries out the function of the system
Has a physical view of the system

SUPERVISORY USER
May or may not have local view
Generally familiar with operation
Driven by budget considerations
Often acts as a middleman between users and higher levels of management

EXECUTIVE USER
Has a global view
Provides initiative for the project
No direct operating experience
Has strategic concerns

Table 3.1: Characteristics of different users
3.1.3 Categorizing users by level of experience

It should be obvious that different users will have different levels of experience; unfortunately, it is common for systems analysts to assume that all users are blithering idiots when it comes to the subject of computers. Perhaps this was a safe assumption ten years ago, but it is likely to get you into a lot of trouble in most organizations today [6]: today, one can distinguish between rank amateurs, cocky novices, and a small (but rapidly growing) number of true computer experts. The amateur user is the one who has never seen a computer and who exclaims loudly and frequently that he or she “doesn’t understand all this computer stuff.” Often, such a user is a middle-aged worker or business person who happily survived 16 years of education and another 10 or 20 years in a job before computers were introduced; however, it is also common to find younger users (those still in their twenties) who find computers boring, intimidating, or irrelevant to their lives.

This presents a challenge to the systems analyst who loves to talk about “on-line access” and “menu-driven human-machine dialogues” or other such terminology — but if the systems analyst does his or her job properly, there is no reason why the user should be interested in or knowledgeable about computers. Indeed, the real problem with the amateur user is somewhat more subtle: he or she may find it difficult to understand the “language” that the systems analyst uses to describe the features, functions and characteristics of the system to be built, even though that language avoids obvious computer-related terminology. As we will see in Parts II and III, the job of systems analysis involves the creation of a number of models of the system to be built.

These models are formal and rigorous representations of a computer system, and at the same time they are abstract representations of the system. Most of the models involve graphics (pictures) supported by detailed text, and the overall representation (which is needed to ensure a formal, rigorous description) strikes some users as overwhelmingly mathematical and thus unreadable. These may be users who remember the difficulty of reading the complex graphical notation used in organic chemistry or the equally complex notation used in differential calculus and algebra. Whatever the reason, the result is the same: quite apart from understanding computer technology, if the user cannot understand the model of the system, there is little chance that he or she will be satisfied with the system when it is finally built. [7]

A second type of user is the one I like to call the “cocky novice,” the person who has been involved in one or two systems development projects, or (even worse) the user who has a personal computer and who has written one or two (ugh) BASIC programs. This user often claims to know exactly what he or she wants the system to do and is prone to point out all the mistakes that the systems analyst made on the last project. This is all fine, except for one thing: the user often becomes far too involved in discussions about the specific technology that will be used to implement the system. Thus, the user may say to the systems analyst, “I need a new order processing system, and I’d like it to be built with a Web front-end connected to our Intranet, and I think we should either use Microsoft Access or Oracle.” These may eventually turn out to be the right technical choices, but it is premature to even consider the hardware, programming language, and database packages before the true requirements of the system have been documented. Indeed, in the extreme case, the user’s “suggestion” about the appropriate hardware and software may turn out to be a “solution looking for a problem,” that is, the discovery that there are underutilized hardware and software resources that can be put to some other use.
There are, of course, some users who really understand systems analysis, as well as the underlying technology of computers (as well as their own business area, too!). It is a pleasure working with these people; indeed, the only problem may be that the user and the systems analyst derive so much pleasure talking about the tools and techniques of systems analysis that they forget that their true objective is to build a functioning system! [8]

3.2 Management

Management is a rather loose term; indeed, the systems analyst is likely to come into contact with several different kinds of managers:

* **User managers** — managers in charge of several people in the operational area where the new system will be used. This was discussed above. These are usually middle-level managers who want systems that will produce a variety of internal reports and short-term trend analyses. The internal reports are usually financial reports, operational reports, exception reports, and the like.

* **IS/IT managers** — the person in charge of the systems development project itself, and the higher-level managers who are concerned with the overall management and allocation of resources of all the technical staff in the systems development organization.

* **General management** — top-level managers who are not directly involved in the IS/IT organization nor in the user organization. This might include the president and/or chairman of the organization, and/or the top financial management (the controller, vice president of finance, etc.). These managers are generally more interested in the strategic planning systems and decision-support systems that were discussed in Chapter 2. While top management does need internal financial reports, they usually don’t need the level of detail (especially in the area of exception reports) that the user managers need. And they focus more attention on external information: government regulations, reports of competition in their marketplace, reports on new markets and products, and so on.

The primary interaction between the systems analyst and all these managers has to do with the resources that will be assigned to the project. It is the systems analyst’s job to identify and document the user’s requirements and the constraints within which the system must be built. These constraints usually consist of resources: people, time, and money. Thus, the systems analyst will eventually produce a document that says, “The new system must carry out functions X, Y, and Z, and it must be developed within six months, with no more than three programmers from the IT department, at a cost of no more than $100,000.” Obviously, management will want an ongoing assurance that the systems development project is staying within these constraints — it is not falling behind schedule or exceeding its budget. But these are issues of project management, not systems analysis. [9] And managers from several different functional areas often form a steering committee that helps prioritize potential development projects so that the most cost-effective projects get done first. There are several points you should keep in mind about managers:

* The higher the level of manager, the less he or she is likely to know or care about computer technology. While this is a generalization, it is usually a fairly safe one with the current generation of senior managers — unless you’re working in a computer company like IBM, Microsoft, etc. This should not affect you as a systems analyst (systems designers have a more difficult job!), but you should remember to concentrate on discussing the essential characteristics of a system when you talk with them.
The goals and priorities of management may be in conflict with those of the users, especially the supervisory and operational users. Management may even impose a system on the users and force them to use it (e.g., if the user organization has been unprofitable or unable to respond to new changes in the marketplace).

A variation on the above theme: management may not provide the resources, funding, or time that the users feel is necessary to build an effective system. It is convenient for the systems analyst and the user to respond to this by saying that management “doesn’t understand,” but it is often a conscious, calculated choice. For more about the politics of resource funding and scheduling, see Appendix B.

The term management implies a homogeneous group of people who all think the same way; the truth, of course, is usually very different. Managers have different views and opinions, and they often have very different goals and objectives. They argue with each other and compete against each other. Hence, it may turn out that some members of management are very much in favor of the new system, while others are dead set against it. Even worse is the benign neglect that befalls some projects; they finally end after years of thrashing about like a fish out of water.

It is also convenient to assume that once management has made up its collective mind about a systems development project, it stays made up. But this is not necessarily so: external forces outside the organization may cause management to speed up the project schedule, or take resources away from it, or abandon it altogether. This often causes enormous emotional distress to those working on the project — including you as the systems analyst!

The relationship between management and your systems development project may depend quite a lot on the overall management structure of your organization, especially the relationship of the systems development activities to the rest of the organization. The classical organizational structure is shown in Figure 3.2(a); note that the entire data processing organization reports to the head of finance and accounting. The reason for this is that most large organizations originally introduced computers to help automate their accounting activities (e.g., payroll, general ledger, and accounts receivable). Beginning in the 1970s, some organizations began to realize that this was a rather lopsided organizational structure: it virtually guaranteed that the data processing function would be biased toward accounting applications and would have little interest or expertise in other parts of the organization. And as automated information processing began to permeate the organization (e.g., in manufacturing, marketing, and engineering), some organizations changed to the organization chart shown in Figure 3.2(b). By having the data processing (or IS/IT, as it is sometimes called) group report directly to the president of the organization, it becomes clear to everyone that data processing is just as critical to the organization’s survival as manufacturing, engineering, sales, and so on. However, by the 1980s, some organizations had begun to find that the MIS department had become an “empire,” with its own priorities and politics; user organizations, meanwhile, were finding that they had an ever-growing backlog of new systems waiting to be developed by the MIS department. This coincided with the introduction and rapid proliferation of cheap, powerful personal computers; thus, some user departments began to feel that they could develop their own systems, without relying on a centralized MIS function. As a result, some organizations now have a structure like that shown in Figure 3.2(c); while there is still a central MIS department for such “classic” applications as payroll and general ledger, much of the departmental processing is done by system development groups within the departments.
Figure 3.2(a): A classical organization chart
If you work in an organization characterized by Figure 3.2(a), you may find that the systems analysts and the users in various other departments are not as good as they could be; indeed, you are likely to find that much of the systems development projects are the “transaction processing” type that one would find in an accounting department. If your organization looks more like the one shown in Figure 3.2(b), then there is a good chance that your systems development group has a reasonable amount of political “visibility” high in the organization; however, you may find that there is growing frustration about the backlog of new systems waiting to be developed. And if you are working in an organization characterized by Figure 3.2(c), you are likely to have much more direct contact with the users of your system; indeed, you may be reporting directly to them. And you are more likely to find yourself working on personal computers and other small networks of computer systems purchased directly by the user department.

3.3 Auditors, quality assurance, and standards bearers

Depending on the size of your project and the nature of the organization you work in, you may or may not have auditors, quality assurance personnel, and/or members of the standards department participating in your project. I have grouped these people into a single category, because their objective and perspective are generally similar, if not the same. The general objective of this group is to ensure that your system is developed in accordance with various external standards (external, that is, to your project): accounting standards developed by your organization’s accounting firm; standards developed by other departments in your organization or by the customer/user who will inherit your system; and possibly standards imposed by various governmental regulatory agencies. In addition, such a group may be concerned with “process management” — i.e., they may want to ensure that your project team has carefully followed the organization’s “official” software development process. [11] There are three problems that you should anticipate when working with auditors, quality assurance people, or members of the standards department:

1. They often don’t get involved in the project until the very end — after the systems analysis, design, and programming work have been finished, and the formal testing activity has commenced. At this point, of course, it is very difficult to make major changes to the system.
2. They are often familiar with an older notation or format for documenting system requirements (e.g., flowcharts). Thus, it is usually very important to ensure that the system models that you develop (peek at Chapter 4 to see some examples!) are understandable. [12]
3. Unfortunately, members of this group are often more interested in form than substance: if your documents are not exactly right, they may be rejected.

3.4 Systems analysts

This is you! The systems analyst is a key member of any systems development project, and the previous sections of this chapter have already given you several examples of how the systems analyst interacts with other players in the project. In a broader sense, the systems analyst plays several roles:

* Archaeologist and scribe — As a systems analyst, one of your main jobs is to uncover detail and to document business policy that may exist only as “tribal folklore,” passed down from generation to generation of users.
Figure 3.2(b): A more current organization chart
* Innovator — The systems analyst must separate the symptoms of the user’s problem from the true causes. With his or her knowledge of computer technology, the analyst must help the user explore useful, new applications of computers — and new ways for the user to conduct business. While many early computer systems merely perpetuated the user’s existing business at electronic speed, systems analysts are being challenged today to help the user find radically new, innovative products and markets made possible with the computer. Edward De Bono’s *Lateral Thinking* (De Bono, 1970) is worth reading for interesting new ways of thinking about problems.

* Mediator — As mentioned earlier in this chapter, it is the systems analyst who often finds himself in the middle of users, managers, programmers, auditors, and various other players — all of whom frequently disagree with one another. There is a temptation for the systems analyst to try to impose his or her own view of what the system should look like or what functions it should contain. But the analyst’s primary job is to achieve a consensus; that requires the delicate art of diplomacy and negotiation!

* Project leader — This is not a universal role, but it happens often enough to mention it here. Since the systems analyst is usually more experienced than the programmers on the project, and since he is assigned to the project before the programmers begin working, there is a natural tendency to assign project management responsibilities to the analyst.

This means that, as a systems analyst, you need more than just the ability to draw flowcharts and other technical diagrams! You need *people skills* to interview users, mediate disagreements, and survive the inevitable political battles that surround all but the most trivial project. You need *application knowledge* to understand and appreciate the user’s business. You need *computer skills* to understand the potential uses of computer hardware and software in the user’s business. And (obviously!) you need a logical, organized mind: you must be able to view a system from many different perspectives; you must be able to partition it into levels of subsystems; and you must be able to think of a system in abstract terms as well as physical terms. [13] Nobody ever said the job was easy!

### 3.5 Systems designers

As we have implied in earlier discussions, the systems designer is the person (or group of people) who will receive the output of your systems analysis work: his or her job is to transform a technology-free statement of user requirements into a high-level architectural design that will provide the framework within which the programmers can work. The nature of this work is discussed in Chapter 22. In many cases, the systems analyst and the systems designer are the same person, or members of the same unified group of people. Even if they are different people, it’s important for the systems analyst and systems designer to stay in close touch throughout the project. The reason for this is the *feedback* that occurs between systems analysis and systems design: the systems analyst has to provide sufficiently detailed information for the systems designer to concoct a technologically superior design; and the systems designer has to provide sufficiently accurate information so that the systems analyst can tell whether the user requirements he or she is documenting are technologically feasible. Based on information received from the systems designer, the systems analyst may have to negotiate with the user to modify user requirements.
Figure 3.2(c): Systems development within user organizations
3.6 Programmers

One could argue that in the best of all worlds there would be no contact between a systems analyst and a programmer. Particularly on large systems development projects, the systems designers are likely to be a “buffer” between the systems analysts and the programmers; that is, the systems analysts deliver their product (a technology-independent statement of the requirements of the system) to the system designers, and the system designers deliver their product (an architectural description of the hardware and software components that will be used to implement the system) to the programmer. There is another reason why the systems analyst and the programmer may have little or no contact with each other: work is often performed in a strictly serial sequence in many systems development projects. [14] Thus, the work of systems analysis takes place first and is completely finished before the work of programming begins. This means that the systems analyst has finished his or her work and has perhaps been reassigned to a new project before the programmer is even brought into the project. However, there is likely to be some contact between programmers and systems analysts for the following reasons:

- On small projects, the roles of analysis, design, and programming are combined, so it may turn out that one person does the work of systems analysis and systems design, and then continues interacting with the programmer. Or it may turn out that one person does the work of systems design and programming.
- The analyst sometimes serves as the project manager, so even though she or he may have finished the work of specifying the requirements of the system, the analyst will still be involved in the project and will have some contact with the programmer.
- It is often the programmer who discovers errors and ambiguities in the “statement of requirements” produced by the systems analyst, for it is programming where, as my colleague Scott Guthery puts it, “the tire meets the road,” where a wishy-washy statement of what the system has to do gets turned into a set of specific, detailed COBOL statements. If something is missing, wrong or confusing, the programmer has two choices: to ask the systems analyst for clarification, or to ask the user. [15]
- As mentioned in Chapter 2, many organizations are now finding it necessary to replace operational systems that were built 20 years ago. In the vast majority of these redevelopment projects, there is virtually no documentation that describes (1) how the system works, or (more importantly!) (2) what the system is supposed to do. And since the systems are 20 years old, there is a whole new generation of users involved; the users who were initially involved in specifying the requirements of the system have retired or quit, and the new generation of users has little idea of the detailed policy requirements embodied in the system. At this point, all eyes turn to the maintenance programmer, who has been keeping the system running for the past several years; this person, too, is likely to be a second- or third-generation worker, having had no contact with the designers and programmers who first constructed the system! As Nicholas Zvegintzov (author of the newsletter Software Maintenance News) points out:
"Up to now, the key computer professional was someone who could learn enough about the needs of organizations to express them in computer language. In the future, as our society becomes irrevocably computerized, the key professional [will be] someone who can learn enough about computerized systems to express them in human language. Without that someone, we [will] have lost control of our society. That someone is the reverse engineer. Software maintainers are the reverse engineers of society."

* Some organizations are beginning to change their project teams from a vertical structure to a horizontal structure. The typical assignment of duties (which is presumed throughout this book) involves all the duties of systems analysis being assigned to one person (or group of people); similarly, all the design activities are assigned to the designer, and all the programming activities are assigned to the programmer. To the extent that this approach is followed, it would certainly seem that systems analysts and programmers would have little contact with one another. But some organizations are beginning to realize that there is an inherent conflict with this approach: systems analysts are usually relatively senior, experienced people in the organization, and yet they are being asked to carry out not only the high-level conceptual statement of system requirements, but also the low-level “nitty-gritty” details of the user’s requirements; a similar conflict exists with the programmers, who are typically more junior and less experienced. One solution is to give the senior technical personnel (whose title happens to be systems analyst) all the high-level activities: high-level systems analysis, high-level design, and the coding of top-level modules in the system; similarly, the junior-level technical people are given low-level, detailed assignments in the analysis area and in the design area, and in the programming area. To the extent that this approach is followed, systems analysts and programmers remain in close contact with one another throughout the project; indeed, each is doing some of the work formerly done by the other. This point is discussed again in Chapter 23.

### 3.7 Web designers

While traditional computer systems are typically implemented in such programming languages as COBOL or C++, many of the new systems today are designed to interact with end-users via the World Wide Web. Thus, while there may still be a great deal of “back-end” processing and database activity, there is a “front-end” user-interface that end-users access via browsers such as Netscape Communicator or Microsoft’s Internet Explorer. This means that the development project is likely to include specialists who not only understand such languages as Java and PERL, but also the appropriate graphics and multi-media tools for creating a user interface that will be effective, enticing, user-friendly, and also efficient.

Some aspects of this technology — e.g., the HTML codes for constructing simple Web pages — are widely understood, and can be implemented by both veteran programmers and relatively non-technical individuals. Thus, while the programmers, designers, and other technical members of the team may be assigned the task of implementing the Web front-end, it’s also possible that members of the user community — e.g., marketing personnel, individuals from the graphics art department, etc. — will get involved.
Other aspects of the Web-related portion of the project are much more technical, and require very skilled personnel. In some cases, most or all of the application processing will take place on the “server” hardware that also handles HTML requests for page displays. This may require the application developers to be familiar with development tools (e.g., ColdFusion) and database technologies (e.g., object-oriented databases such as Computer Associates’ Jasmine) that are radically different than the older technologies that run on the back-end mainframes and client-server architectures. And because the Web is intimately connected to the Internet, there’s a good chance that the project team will require specialists who are familiar with networking, security, and other technical aspects of the Internet.

### 3.8 Operations personnel

Just as one could argue that the systems analyst would never encounter a programmer, it could be argued that the systems analyst need never have any interactions with the operations personnel who are responsible for the computer center, telecommunications network, security of the computer hardware and data, as well as the actual running of computer programs, mounting of disk packs, and handling of output from computer printers. All this happens after a new system has not only been analyzed and designed, but has also been programmed and tested.

However, there is more to this than meets the eye: the systems analyst must have some understanding of the constraints imposed on a new system by the operations personnel, for this becomes part of the detailed specification produced by the systems analyst. That is, the systems analyst may produce a document that says, “The new order system must be capable of carrying out functions X, Y, and Z — and, in order to conform to the requirements of the Operations Department, it must occupy no more than 16 megabytes of memory on the mainframe computer. The system must be implemented using PC-compatible workstations running Microsoft Windows operating systems, connected to the company’s XYZ telecommunications network.”

In some cases, the operational details of the system may be a matter of negotiation between the user and the central computer operations group. This is especially common today, since users are often in a position to acquire their own personal computers or department-sized minicomputers. While many of these computers can be operated by clerical or administrative people in the user organization (thus not requiring the specialized talents of the operations personnel), and while many of the computers can operate in a normal office environment (thus not requiring the special wiring and air-conditioning equipment typical of large mainframe computers), it is still generally true that these small machines will have to communicate with the mainframe computer (e.g., to download part of a corporate database, or to upload the results of departmental computing), and it is often true that the small PCs and/or minicomputers have a need to communicate with one another through a local area network or some other telecommunications facility. All this usually involves interaction with the operations personnel; only a truly independent stand-alone system can be built without their assistance and approval. These operational issues are documented in a part of the systems analysis effort known as the user implementation model. This is covered in detail in Chapter 21.
3.8 Summary

As we have seen in this chapter, the systems analyst is an orchestrator, a communicator, and a facilitator. It will become evident in Parts II and III that the systems analyst does a great deal of work on his or her own, but even more work is done in harmony with the other players in the systems game. As a systems analyst, the more you know about the people you will be working with, the better off you will be. All the players are people; and they have different goals, different priorities, and different perspectives. Though they may be publicly committed to its success, they may have hidden agendas that are opposed to one or more aspects of the project. The questions and exercises at the end of this chapter are intended to make you think more about these issues. For additional information, consult Block's excellent book on project politics (Block, 1982) or even Sun Tzu's classic book on the art of war (Tzu, 1983).
References

1. A common situation of this nature is the contracting officer in a governmental organization. In most cases, this person is not the user, and may not know very much about the real needs of the user, but he or she is the one designated to maintain all official communication with the person (or company) developing the system.

2. There are variations on this terminology; (Teague and Pidgeon, 1985), for example, refers also to the “beneficial user,” the user who will receive the benefits of the new system. This person may not have any direct contact with the system, but will profit in some way from the improved service or functionality of the new system.

3. There are related issues that emphasize the fact that the new system is part of an even larger system: the user will ask, “Will I get back strain or tendonitis from sitting in front of a terminal all day?” “Do I have to worry about radiation leakage from the CRT screen?” “What if I don’t know how to type?” And, most important, “What if this new system takes over my job and puts me out of work?”

4. In the extreme case, this also means that it is the operational user who can make or break a new system. They may seem passive, and they may not have the power or authority to approve a systems development project, but if they sabotage it or simply fail to use it, the new system will have failed.

5. Note that this is a feature of an operational system (as we defined the term in Chapter 2), but generally not a feature of the decision-support systems. And note also that higher-level managers are generally more interested in systems that provide them with a competitive advantage than the system that reduces the operational staff by one or two people.

6. Even if every user you encounter is unaware of and uninterested in computer technology, you should avoid the common mistake of treating them as a subhuman form of life. Young systems analysts and computer programmers, especially the “hackers” who began playing with computers when they were in elementary school, assume that everyone should be fascinated and facile with computers, and that those who are not are either (1) mentally deficient or (2) members of an older generation, and thus unworthy of any respect or consideration. Meanwhile, the world is full of users who don’t like computers for a variety of legitimate reasons, and there are users who are far too busy being an expert in their own profession or business to worry about being an expert in computers. They have the same opinion of computer programmers and systems analysts that they have of electricians, carpenters, plumbers and auto mechanics: a healthy respect for the expertise and craftsmanship required of the job, but a total lack of interest in the details. Understanding this point will, in many cases, determine whether you succeed or fail on your first few projects as a systems analyst.
7. An analogy: if you were going to have a house built for you, you would begin by talking with an architect about the desired features of the house. After a great deal of discussion, the architect would retreat to his office and then eventually come back to you with a number of drawings and/or scale models of the house. If you refused to look at the drawings or objected that they were “too mathematical,” the architect’s chances of success would be small indeed. What you would probably do is take the architect to an existing house and say, “Build me one like that!” Unfortunately, we are not often in a position to do that in the computer field, though prototyping is sometimes a viable way of accomplishing the same thing.

8. It is also encouraging to see that more and more of these “experts” are moving into top management positions in business organizations and top leadership positions in other parts of our society. Citicorp and American Airlines, not to mention a number of computer companies and other high-tech organizations, are run by people who rose up through the ranks of data processing. And, by the mid-1980s, there were approximately half a dozen members of Congress who are former programmers and systems analysts.

9. However, sometimes the systems analyst is very involved in project management. We will discuss this point in more detail in Chapter 16, as well as in Appendix B.

10. We will discuss the backlog of applications in more detail in Chapter 6.

11. This is common, for example, in organizations striving for ISO-9000 certification, and for those seeking a certification of their software “maturity” from the Software Engineering Institute.

12. However this is changing. For example, many of the Big 6 accounting firms in the United States are now quite familiar with the structured analysis documentation tools described in this chapter; hence they should have no problem participating in one of your projects as an auditor.

13. Indeed, it is because of this requirement for expertise in many areas that most computer scientists feel that artificial intelligence and expert systems won’t be able to be applied to systems analysis for several more years. More on this in Chapter 25.

14. We will discuss some alternatives to this sequential approach, particularly those known as evolutionary development, or fast tracking, in Chapter 5. Indeed, for some projects systems analysis continues while programming is going on.

15. Indeed, direct contact between the programmer and the user is more common than you may think. In many cases, the systems analyst never gets around to describing the low-level details of the system, and the high-level users with whom the systems analyst communicates may be unaware of or uninterested in those details. Thus, it often turns out that the programmer must talk directly to the low-level user to find out exactly what the system is supposed to do. This is significant, because many organizations bemoan the fact that 50% of their systems development projects are spent on testing; in fact, it may well turn out that the work taking place under the official guise of testing is, in fact, systems analysis work that could have (and probably should have) taken place earlier in the project.
Questions and Exercises

1. List at least one additional player that you would expect to interact with in a systems development project.

2. Describe a project in which the systems analyst did not have direct contact with the real user. What were the advantages and disadvantages of this situation? What alternative arrangements could have been made?

3. Can you think of another term for user besides owner or customer?

4. Can you think of any situation where the systems analyst should not talk to the user?

5. What are the advantages and disadvantages of having the user be a full-time member of the systems development project team? Can you think of any specific projects where it makes particularly good sense to have a user on the project team?

6. What are the advantages and disadvantages of having the user be the manager of the systems development project team? Can you think of any specific projects where it would make particularly good sense to have the user manage the project?

7. What are the advantages and disadvantages of having the user develop an information system entirely by himself? Can you think of any projects where it makes particularly good sense to have the user be the analyst, designer, programmer, and manager?

8. How much should a user know about computers and software in order to participate in a project team during the systems analysis phase? How much should he or she know about the tools and techniques of systems analysis?

9. How much should a user know about computers and software in order to manage a systems development project team? How much should he or she know about systems analysis in order to be an effective manager?

10. How much should a user know about computers and software in order to accomplish a systems development project entirely by himself? How much should he or she know about systems analysis?

11. What special precautions would you take as a systems analyst if you did not have direct contact with the user? Do you think that the modeling tools described in this book would be sufficient?

12. Section 3.1.2 lists several concerns that an operational user might have about a new system. List three more likely concerns. Do you think these are reasonable concerns, or do they just reflect the typical user’s unfamiliarity with computers?

13. What moral or ethical responsibility does the systems analyst have to the operational user if it she or he is convinced that it won’t cause layoffs, but the user is concerned that it will? (See also question 19.)

14. Describe a scenario where the operational users could cause a new system to fail. Do you think your scenario is realistic? Couldn’t the supervisory user simply mandate that the system be used?

15. When do you think the human interface issues should be discussed with the users? Early in the project? Late? What are the trade-offs? (You’re allowed to peek ahead at Chapter 21 if you wish.)
Questions and Exercises (cont.)

16. Do you think it’s unrealistic that operational users would have only a local view of the system in which they participate? Do you think it is safe for the systems analyst to take this for granted? Do you think this is a good situation? Should the systems analyst try to provide a global view — the “big picture” — to the operational users?

17. Give an example of a physical, or implementation-oriented, view of a system that an operational user might have. Do you see any problems with this?

18. What should the systems analyst do if the supervisory user won’t let him or her talk directly to the operational users? How can the systems analyst deal with this situation?

19. What moral or ethical responsibility does the systems analyst have to the supervisory user if the operational users express concern about possible layoffs caused by the new system? (See also question 13.)

20. Give an example of a system where the supervisory user may not be familiar with the detailed business policy currently being carried out by the operational users.

21. Why are executive-level users typically not interested or concerned about the possible savings to be achieved by personnel reductions (e.g., through layoffs or attrition) made possible by a new system?

22. How closely involved should the executive-level users be in the development of a new information system?

23. What options does the systems analyst have if the user doesn’t understand abstract, paper models?

24. How should the systems analyst deal with the “cocky novice” described in this chapter? What if the user insists on a particular choice of computer hardware or software for the new system?

25. How much responsibility should the systems analyst take for gaining consensus among the users? What if the analyst fails to do this?

26. What risks do you think the systems analyst faces from management, as discussed in Section 3.2? What can the systems analyst do to minimize the risks?

27. What should the systems analyst do if management’s goals and priorities are in conflict with those of the user?

28. When do you think operations people should get involved in a project?

29. Should systems analysis and systems design (and programming, too) be done by the same person (or cohesive group of people)? What are the advantages and disadvantages?