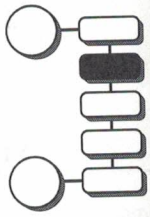
CHAPTER 4

FUNCTIONS AND SPECIFICATIONS



How can I express what the client wants in engineering terms?

UP TO now we have focused on understanding what the client and users want and need from a design. In this chapter we move from using the language of the client to the language of the engineer. In particular, we want to translate the client and user needs and desires into terminology that helps us realize those needs and measure how well we meet them. In the terminology of engineers and designers, we identify functions and performance specifications. These three terms represent related, yet distinct aspects of how a designed artifact does what it was designed to do.

First we will look at *functions* and functional specifications because they tell us *what the designed object must do* to realize the stated objectives. We have to establish *which* functions have to be performed before we can specify how well the functions must be performed. Then we will consider *performance specifications* that tell us *how well* the designed object must do something.

The ordering of functions, specifications, and metrics (introduced in Section 3.4) is somewhat arbitrary. For example, in many cases a designer will consider how the realization of a particular objective might be measured before she determines precisely the function to be performed. In other cases, some of the performance specifications may have been set out by the client when the design job begins. We present these design tools and techniques in this particular order because it delineates a logical way of thinking about them. However, the iterative nature of design inevitably produces variation in the timing of when functions, performance specifications, and metrics are introduced and processed.

4.1 IDENTIFYING FUNCTIONS TO REALIZE OBJECTIVES

If a child is asked what a bookcase does, he might answer that "It doesn't *do* anything, it just sits there." An engineer, however, would say that the bookcase does a number of different things, and that it must do them well to be a successful design. In this view the bookcase resists the force of gravity exactly, so that books neither fall to the floor nor are forced into the air. The shelves of the bookcase may have dividers, to aid in separating the books into categories chosen by the owner. And, if the bookcase was designed with an eye to interior

design, we might say that it enhances the visual appeal of the room. Each of these ways that our designed bookcase does things are functions. An engineer looking at designed objects is educated to see that *artifacts do things*, even when they "just sit there." Understanding what a designed object must do is essential to creating a successful design. In this section we will look more deeply at what we mean when we talk about a design doing something, and then we will look at techniques for understanding and listing functions.

It is particularly important for a designer to be able to properly specify functions because there are consequences for the engineer when he fails to understand and design for *all* of the functions in a design. The literature of forensic engineering is rich with cases in which engineers failed to realize some additional function(s) that had not been met, often with tragic results.

4.1.1 What are functions?

We can think of functions in a number of different ways. In elementary calculus, for example, we say that y, a dependent variable, is a *function* of x, a single independent variable, when we write $y = f(x)$. That is, the value of y depends on the value of x. In multivariate calculus we extend this notion to include multiple independent variables and multiple dependent responses. Management studies refer to *transformation functions* in which a vector of inputs (labor, materials, technology, etc.) is transformed into a set of outputs (products, services, etc.). In each of these cases we are highlighting the existence of a relationship between some independent variables (i.e., *inputs*), and response or dependent variables (i.e., *outputs*), and we are characterizing that relationship in a formal way. We use a similar notion of functions when we consider the functions of an object we are designing. To designers, functions are the things that the designed object must do in order to be successful. As such, the statement of a function usually consists of an "action" verb and a noun. For example, lift, raise, move, transfer, or illuminate are action verbs. The noun in the statement of function may start off as a very specific reference, but experienced designers look for more general cases.

Functions are often expressed as verb-object pairs.

For example, one of the bookcase functions was to resist forces of gravity, which we might characterize as "support books." However, this would imply that the bookcase would only be used to hold books. But shelves in bookcases often support trophies, art, or even piles of homework. Thus, a more basic and more useful statement of the function to be served here is that shelves resist the force of gravity associated with objects weighing less than some predetermined weight. That is, our statement of function is that shelves will support some number of kg. (or lbs.). When describing functions, then, we should use a verb-noun combination that best describes the most general case.

We also want to avoid tying a function to a particular solution. If we were designing a cigarette lighter, for example, we might be tempted to consider "applying flame to tobacco" as a function. This implies that the only way to light the tobacco is by using a flame (and that tobacco is the only material to be lit). Car lighters, however, use electrical resistance in a wire to achieve this function. Thus, a better statement of this function might be "ignite leafy matter," or even "ignite flammable materials." (Somewhat parenthetically, we could consider the following questions. In light of the well-documented health hazards associated with smoking, is there an ethical issue for an engineer who is asked to design a better cigarette lighter? Is it an appropriate design task? We discuss ethics in engineering

and design in Chapter 9, but we note here that the issue may not arise if the lighter is viewed as a camping tool.)

We can also categorize functions as being either *basic* or *secondary* functions. A *basic function* is defined as "the specific work that a project, process, or procedure is designed to accomplish." *Secondary functions* would be (1) any other functions needed to do the basic function or (2) those that result from doing the basic function. Secondary functions can themselves be either required or unwanted functions. *Required secondary functions* are clearly those needed for the basic function. Consider, for example, an overhead projector. Its basic function is to project images, and it has required secondary functions that include converting energy, generating light, and focusing images. *Unwanted secondary functions* are undesirable byproducts of other (basic or secondary) functions. For the overhead projector these include generating heat and generating noise. Such undesirable byproducts often generate new required functions, such as quieting noise or dissipating generated heat.

4.1.2 How do we identify and specify functions?

How do we identify and specify the functions to be performed by the artifact that is the focus of our particular design project? We must determine these functions in order to ensure that our final design does what it is supposed to do. Thus, we now describe four methods used to determine functions: enumeration, analysis of "black" and "transparent" boxes, construction of function-means trees, and reverse engineering, a method that is also known as dissection.

4.1.2.1 Enumeration The most basic method of determining functions for a designed object is to simply *enumerate* or list all of the functions that we can readily identify. This is an excellent way to begin functional analysis for many objects. It leads us to consider what the basic function of the object is and it may prove useful for determining secondary functions. However, we might get "stumped" very early in this process. Consider, for example, a bridge. If the bridge is used for highway traffic, we might note that its basic function is to act as a conduit for cars and trucks, and then we might scratch our heads before being able to add much to this initial, single-entry list. However, there some useful "tricks" we can use to extend an enumerated list.

Effective function enumeration goes beyond making a list.

One trick is to imagine that an object exists and ask what would happen if it suddenly vanished. If a bridge disappeared entirely, for example, any cars on the bridge would fall into the river or ravine over which the bridge crosses. This suggests that one function of a bridge is to support loads placed on the bridge. If the abutments ceased to exist, the deck and superstructure of the bridge would also fall, which suggests that another function of the bridge is to support its own weight. (This may seem silly until we recall that there have been more than a few disasters in which bridges collapsed because they failed to support even their own weight during their construction. Among the most famous of such infelicitous bridges is the Quebec Bridge over the St. Lawrence River, which collapsed once in 1907 with the loss of 75 lives and again in 1916 when its closing span fell down.) If the ends of a bridge that connect to various roads disappeared, traffic would not be able to get on the bridge, and any vehicles on the bridge would be unable to get off. This suggests that another function of a bridge is to connect a crossing to the road network. If road dividers on our

bridge were removed, vehicles headed in one direction can collide with vehicles headed in the other. Thus, separating traffic by direction is a function that many bridges serve, and it is a function that can be accomplished in several ways. For example, New York's George Washington Bridge assigns different directions of traffic on each its two levels. Other bridges use median strips.

Another way to determine functions is to consider how an object might be used and maintained over its lifetime. In the case of our bridge, for example, we might note that it is likely to be painted, so that one function is to provide the bridge's maintenance workers with access to all parts of the structure. This function might be served with ladders, catwalks, elevators, and so on.

Consider once again our beverage container design problem. Here, because we have ample experience with such containers, we can readily name or list the functions served by a beverage container, including at least the following:

- contain liquid
- get liquid into the container (fill the container)
- get liquid out of the container (empty the container)
- close the container after opening (if it is to be used more than once)
- resist forces induced by temperature extremes
- resist forces induced by handling in transit
- identify the product

Note that the functions of getting liquid into and out of the container are distinct. This is evident after a brief reflection on canned beverages: Liquid is sealed in by a permanent top, while access is obtained through a pull tab. We might have noticed this distinction between the filling and emptying functions had we considered the "life cycle" of a beverage container.

At the heart of our approaches to function enumeration lies the need for the designer to list the verb-noun pair that corresponds to each and every function of the designed object. However, since enumeration is often difficult, we must turn to other methods.

Black box analyses connect outputs to inputs.

4.1.2.2 Black boxes and transparent boxes Recall that our earlier discussions of mathematical and management functions featured inputs and outputs, both singly or in groups. This input-output model is also useful in modeling system designs and their associated functions. One tool that helps relate inputs and outputs, and the transformations between them, is the *black box*. A black box is a graphic representation of the system or object being designed, with inputs shown entering the box on its left-hand side and outputs leaving on the right. *All* of the known inputs and outputs should be specified, even undesirable byproducts that result from unwanted secondary functions. In many cases functional analysis helps us identify inputs or outputs that have been overlooked. Once a black box has been drawn, a designer can ask questions such as "What happens to this input?" or "Where does this output come from?" We can answer such questions by removing the cover of the black box, thus making it into a *transparent box*, to see what is going inside. That is, we expose transformations from inputs to outputs by making a box transparent. We can also link more detailed "subinputs" to (smaller) internal boxes within any given box that produce related "suboutputs."

Consider, for example, a system with three inputs: an airborne signal within that part of the frequency spectrum containing the radio frequencies (RF), a controllable source of electrical power, and a vector of desired outputs (such as particular stations and volume levels). This system has three obvious outputs: sound, heat, and a display that indicates how the user's desired frequency and volume level were realized. This system, the radio, can be viewed as being contained in a box that transforms an RF signal into another signal that we hear the audible sounds of music, talk, and perhaps noise! How does this happen? What functions are performed in a radio? Can we identify the many functions of a radio by looking inside the radio box?

We model the functions of our radio with the series of boxes shown in Figure 4.1. In Figure 4.1(a) we see the radio's most basic function of converting an RF signal to an audio signal is achieved while several byproducts, both desired and undesired, are generated at the same time. If we take the cover off this box (Figure 4.1(b)), we see several new black boxes within. These boxes include transforming the power from that of a wall outlet, 110V, to a level appropriate for the radio's internal circuitry, probably 12V. Other internal functions include filtering out unwanted frequencies, amplifying the signal, and converting the RF to an electrical signal that drives speakers. Thus, making our black box cover transparent revealed a number of additional functions. If we had to design the radio, we would probably remove the covers of even more of the boxes we now see. If, on the other hand, we were assembling a radio from known parts, we might stop at this level. This method of making internal boxes transparent and analyzing their internal functions is also called the *glass box method.* No matter which term we use, the effect is the same: We keep opening internal boxes until we fully understand how all inputs are transformed into corresponding outputs and what additional side effects are produced by these transformations.

Black boxes become glass boxes when we ask *how* inputs produce outputs.

The black box method can be a very effective way to determine functions, even for systems or devices that do not have a *physical* box or housing. The only requirement for using a black or transparent box is that *all* of the inputs and outputs be identified. For example, to design a playground for a rainy climate, our inputs would include children, their parents or caregivers, and the rain. Our outputs would include entertained children, satisfied parents, and water. If we forget the water, our playground design will suffer for lack of proper drainage. (As an aside, it is generally not sufficient to include general terms such as "weather" unless we are willing to consider how the weather is translated into water, ice, wind, and heat *within* our box.)

A final point on the black or glass box method is that we must be very careful when we define the *boundaries* or limits of the device or system for which we are identifying functions. These boundaries require a tradeoff. If they are too wide, we may specify functions that are beyond our control, for example, specifying the household electric current for the radio. If

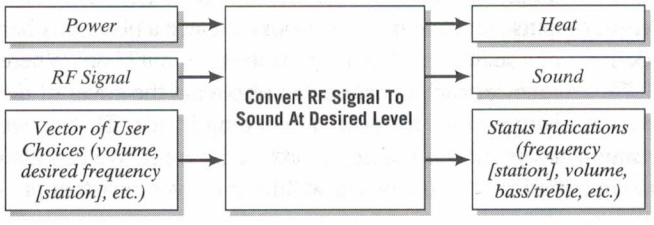


FIGURE 4.1 (a) This is a black box for the radio. Notice that all the inputs and outputs are somehow related in the single, top-level function. We call this top-level function a *basic function.* If we want to know how the inputs are actually transformed into the outputs, we need to remove the cover on the black box.

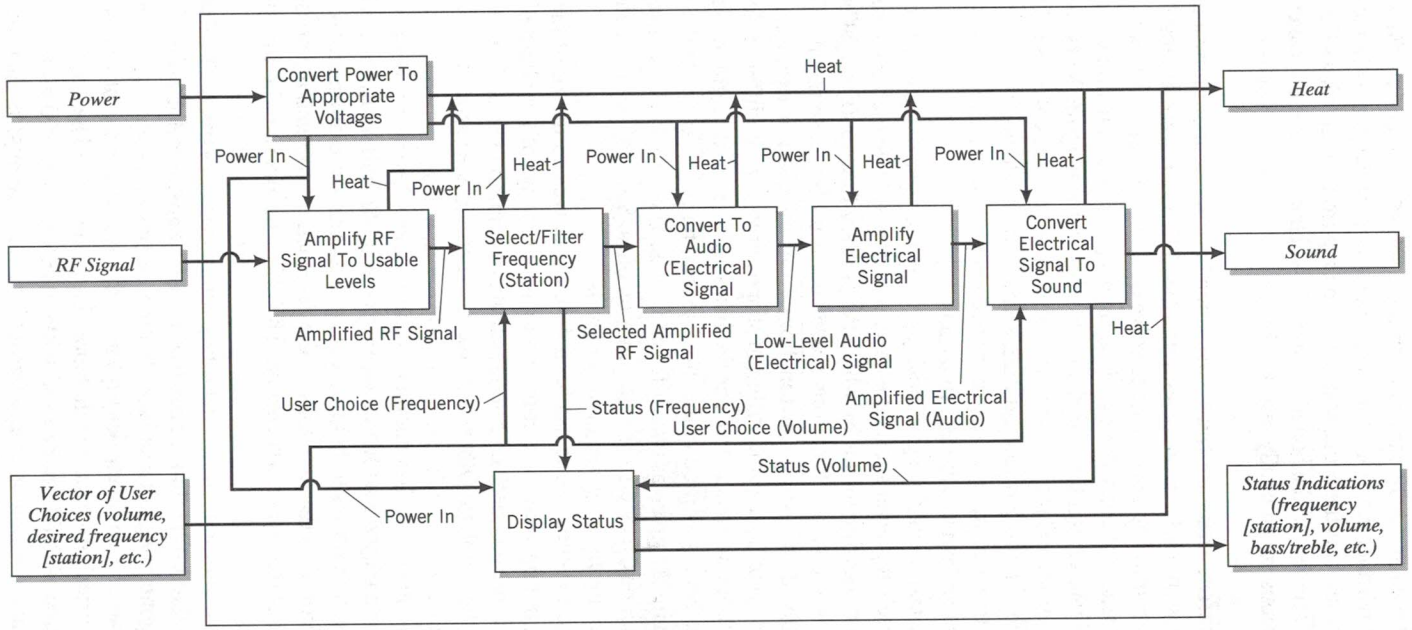


FIGURE 4.1 (b) The cover on the black box has been removed, or made transparent. Notice that in order to transform the inputs into outputs, a large number of secondary functions are required. If our design responsibilities (or our curiosity) demanded it, we could also remove the covers on some or all of these functions. Notice also that this design calls for the heat to be allowed out of the box on its own. In many designs, we would add a specific function, "dissipate heat," and then decide on a strategy for doing so.

boundaries are drawn narrowly, they may limit the scope of the design. For example, it may be an open issue whether the radio output is an electrical signal that is fed to the speakers or whether the radio output is the acoustical signal coming from the speakers. The question being decided by the placement of the boundary, then, is whether or not the speakers are part of the radio. Such decisions are generally resolved by the client and potential users.

4.1.2.3 Function-means tree We often have ideas about how a designed device or system might work early in the design process. While we warn against "marrying your first design" and caution against trying to solve design problems until they are fully understood, it is often true that early design ideas suggest different functional aspects. Consider the hand-held (cigarette) lighter. Clearly, if we use a flame to ignite leafy materials we will encounter different secondary functions than if we use hot wires or lasers. One such difference could be the shielding of the igniting element if the device is to be pocket-sized. A function-means tree can help us sort out secondary functions in cases where means or implementations can lead to different functions.

A *function-means tree* is a graphical representation of a design's basic and secondary functions. The tree's top level shows the basic function(s) to be met. Each succeeding level alternates between showing means by which the primary function(s) might be implemented and displaying the secondary functions made necessary by those means. Some graphical notation is employed to distinguish functions from means. For example, functions and means can be shown in boxes with different shapes or written in different fonts. Figure 4.2 shows part of a function-means tree for the hand-held cigarette lighter. Note that the top level function has been specified in the most general terms possible. At the second level, a flame and a hot wire are given as two different means. These two means imply different sets of secondary functions, as well as some common ones. Some of these secondary functions and their possible means are given in lower levels.

Once a function-means tree has been developed, we can list all of the functions that have been listed, noting which are common to all (or many) of the alternatives and which are particular to a specific means. Functions that are common to all of the means are likely to be inherent to the problem. Others are addressed only if the associated design concept is adopted after evaluation.

A function-means tree has another useful property because it begins the process of associating what we must do with how we might do it. We will return to this issue in Chapter 5 when we present a tool to help us generate and analyze alternatives. That tool, the morphological chart, lists the functions of the designed artifact and the possible means for realizing each function in a matrix format. The effort we put into the function-means tree really pays off then.

Beware of using function-means trees to reinforce preconceived ideas.

Two cautions should be noted regarding function-means trees. The first, and perhaps most obvious, is that a function-means tree *is not* a substitute for either formulating the problem or for generating alternatives. It may be tempting to use the outcome of the function-means tree as a complete description of the available alternatives, but doing so will likely restrict the design space much more than needs be. The second caution is that function-means trees should not be used without using some of the other tools described above. One mistake commonly made by novices (or students) is that they adopt a tool because it somehow "fits" with their preconceived ideas of a solution. This transforms the design process from a creative, goal-oriented activity into just a mechanism for making

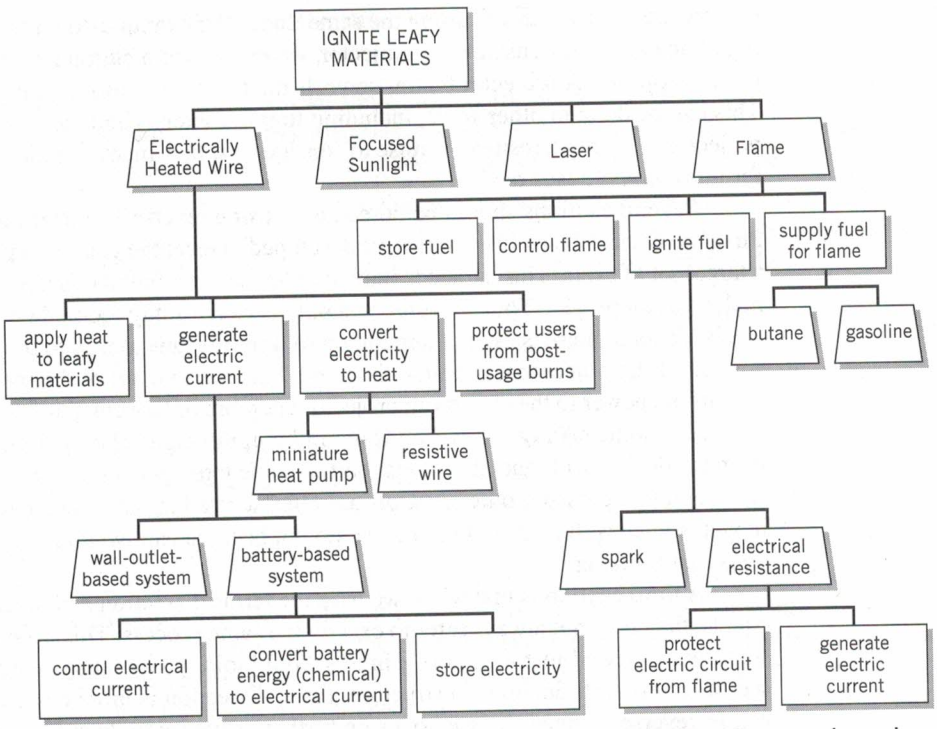


FIGURE 4.2 Part of a function-means tree for a cigarette lighter. (Functions are shown in rectangles, while means are shown in trapezoids.) Notice that there are different subfunctions that result from different means. It is often the case that conceptual design choices will result in very different functions at the preliminary and detailed design stages.

choices that a designer wanted. That is, because the function-means tree allows us to work with appealing means or implementations, we may overlook functions that might have turned up with a less "solution-oriented" technique.

4.1.2.4 Dissection and reverse engineering Most engineers, and indeed, most curious people, ask the question "What does this do?" when confronted with a button, knob or dial. A natural follow-up may take the forms "How does it do that?" or "Why would you want to do that?" When we follow up these questions with remarks on how we might do it better or differently, we are engaging in the art of *dissection* or *reverse engineering*. Reverse engineering means taking an artifact or device that does some or all of what we want our design to do and dissecting—or deconstructing or disassembling—it to find out, in great detail, just how it functions or works. We may not be able to use that design for any number of reasons: It may not do all the things we want, or do them very well; it may be too expensive; it may be protected by a patent; or it may be our competitor's design. But even if all of these reasons apply, we often can gain insight into our own design problem by looking at how other people have thought about the same or similar problems.

The process is actually quite simple. We begin with a means that has been used by a previous designer, then determine what functions are realized by that given means. We then

explore alternative ways of doing the same thing. For example, to understand the functioning of an overhead transparency projector, we might find a button that, when pushed, turns on the projector. A projector button controls the function of turning the projector on or off. This can be done in other ways, including toggle switches and bars along the front of the projector. It is an interesting exercise to consider just how many functions can be thought of for this commonplace device.

Several cautions should be noted about using reverse engineering to find functions. First, the devices being dissected were developed to meet the goals of a particular client and a target set of users. This audience may have had very different concerns than are called for in the current project. Thus, a designer should be sure to stay focused on the current client's needs. Second, there is often a temptation to limit the new means to those that work in the context of the object that is being dissected. For example, all of the means for turning on and off the power to the classroom projector are more or less compatible with a stand-alone device. In some settings, however, it might be more appropriate to remove these controls from the device itself and make them part of some more general room controls. In theaters, for example, lights and other controls are often located in the projection room, rather than in wall switches. It is important that we do not become captive to the design being used to assist our thinking.

A third caution is that while we treat the terms dissection and reverse engineering as equals, they may not always refer to exactly the same process. This is because dissection is sometimes viewed just as it is in a high school biology laboratory, wherein a frog is dissected to reveal its anatomical structure. Here, dissection is more descriptive than analytical. In reverse engineering we go a step further as we try to determine means for making functions happen, which means that we are trying to analyze both the functional behavior of a device and how that functional behavior is implemented.

Dissection should enhance our appreciation of the ideas of others.

There is a fourth consideration, one that we stated earlier but reiterate here. We need to define functions in the broadest possible terms and only focus down when it is necessary. Restricting functions to the most immediate terms found on the object being reverse engineered may lead us to mimic someone else's design, rather than fully appreciating opportunities for new ideas. Further, there are clearly serious intellectual property and ethical issues tied to reverse engineering. It is never appropriate to claim as our own the ideas of others. In some cases this can be a violation of law. We will discuss intellectual property in Chapter 5 and ethics in Chapter 9, but it is always important to respect the ideas of others at least as stringently as any other (tangible) property they might hold. After all, wouldn't we want the same protections for our own ideas?

4.1.3 A repeated caution about functions and objectives

It is often the case that novice designers make lists of objectives when functions are called for, and vice versa. This happens because two new concepts—objectives and functions—are being learned, and the tools used to determine objectives and functions, respectively, are sufficiently similar that they are often confused. The "newness" problem will likely be overcome only by listing examples of each concept and discussing them with team members, teachers, and experienced designers, hoping to obtain in this way a practical feel for both objectives and functions. The confusion related to the tools themselves can be eased

by keeping in mind whether the immediate focus is on "being" terms or on "doing" verbs. As we noted in Section 3.1.2:

Objectives are adjectives; functions are verbs.

- Objectives describe what the designed artifact will be like, that is, what the final object will be and what qualities it will have. As such, objectives detail attributes and are usually be characterized by present participles such as "are" and "be."
- *Functions* describe what the object will *do*, with a particular focus on the input-output transformations that the artifact or system will accomplish. As such, *functions transform inputs to outputs*, and are usually characterized by active verbs.

The distinction between objectives and functions is terribly important, but its centrality is often only grasped fully after a great deal of serious practice.

4.2 DESIGN SPECIFICATIONS: EXPRESSING ATTRIBUTES AND FUNCTIONAL BEHAVIOR

In Chapter 1 we mentioned that design specifications articulate in various ways the attributes and behaviors of a design. Such specifications also provide a basis for evaluating a design because such "specs" become the targets of the design process against which we measure our success in achieving them. Design specifications or requirements are presented in three forms that represents three ways of formalizing what the client or user wants in terms suitable for engineering analysis and design:

Prescriptive specifications specify ***values*** *for attributes of the designed object*. For example, "A step on a ladder is safe if it is made from Grade A fir, has a length that does not exceed 20 in., and is attached in a full-width groove slot at each end."

Procedural specifications identify *specific* ***procedures*** *for calculating attributes or behavior*. For example, "A step on the ladder is safe if its maximum bending stress is computed from $\sigma_{max} = Mc/I$ and is such that σ_{max} does not exceed σ_{allow}."

Performance specifications identify ***performance levels*** that signify the achieved *desired functional behavior*. For example, "A step on a ladder is safe if it supports an 800 lb. gorilla."

Thus, *prescriptive* specifications specify values of attributes that a successful design must meet (e.g., a chicken coop should have a clear plastic roof). *Procedural* specifications mandate specific procedures or methods to be used to calculate attributes or behavior (e.g., the ability to house chickens safely might be specified by requiring application of the USDA poultry guidelines.) .

Performance specifications characterize the desired functional behavior of the designed object or system (e.g., a chicken coop should house 25 chickens). As noted in Section 4.1, determining what a designed object or system must do is essential to the design process. Functional specifications don't mean much if we don't consider *how well* the design must perform its functions. For example, if we want a device that produces musical sounds, we should specify how loudly, how clearly, and at what frequencies the sounds are produced. Thus performance or functional requirements must be specified or defined.

In addition, if a system or artifact has to work with other systems or artifacts, then we must specify how those systems interact. We call these particular specifications *interface performance specifications*.

4.2.1 Attaching numbers to design specifications

It is normally up to the designer to cast functions in terms that facilitate the application of engineering principles to the design problem at hand. Thus, the designer has to translate functions into measurable terms in order to be able to develop and assess a design. We must find a way to measure the performance of a design in realizing a specific function or objective and then establish the range over which that measure is relevant to the design. And designers must determine the extent to which ranges of improvements in performance really matter.

Determining the range over which a measure is relevant to a design and deciding how much improvement is worthwhile are interesting problems. Our conceptual starting point for assessing the value of a gain in performance of a design (at some unspecified cost) is the curve shown in Figure 4.3. It is similar to what economists call a *utility plot* with which the benefit of an *incremental* or *marginal* gain in performance can be found. The utility or value of that design gain is plotted on the ordinate on a normalized range from 0 to 1. The level or "cost" of the attribute being assessed is shown on the abscissa. For example, consider using processor speed as a measure of a laptop computer performance. At processor speeds below 100 MHz, the computer is so slow that a marginal gain from, say, 50 MHz to 75 MHz, provides no real gain. Thus, for processor speeds below 100 MHz, the utility is zero. At the other end of the utility curve, say, above 5 GHz, the tasks for which this computer is designed cannot exploit additional gains in processor speed. For example, browsing the World Wide Web may be more constrained by typing speed or communication line speeds, so that an incremental gain from 5 GHz to 5.1 GHz still leaves us with a normalized utility of 1. Thus, the utility plot is *saturated* at high speeds.

What happens at performance levels between those that have no value and those on the saturation plateau, say between 100 MHz and 3 GHz for the computer design? In this range we expect that changes do matter, and that increases in processor speed do improve the incremental or marginal gain. In Figure 4.3 we show an *S*- or Saturation-curve that dis-

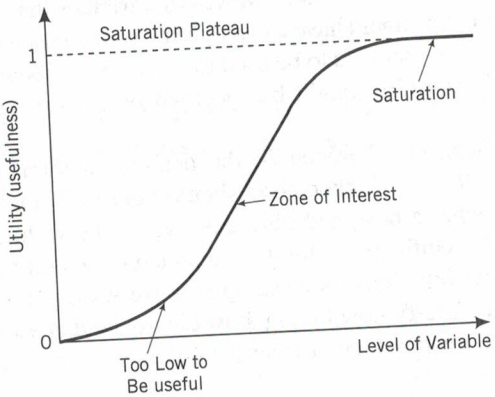


FIGURE 4.3 A hypothetical performance specification curve. Notice that until some minimal level is realized, no meaningful benefit is achieved. Similarly, above some saturation plateau, there is no meaningful benefit in gaining still more. The actual shape of the curve is likely to be uncertain in most cases.

plays *qualitatively* what is happening. There are clearly gains to be made as we move toward faster speeds, and the value of those gains can be determined from the curve. Thus, the utility of the entire S-curve is initially flat (or 0) at low processor speeds, increases measurably over a range of interest, and then plateaus at 1 because added gains are not valued.

This sort of behavior seen in a utility plot is rather common. Economists refer to the *law of diminishing returns*, however, it isn't a "law." We don't usually know the actual shape or precise details of the S-curve (it may not look nearly as smooth as what we have sketched in Figure 4.3), so we choose to approximate it by a collection of straight lines, such as that shown in Figure 4.4. Here there are still regions where gains no longer interest us, as indicated by the horizontal lines at levels 0 and 1. In the middle range, however, we suppose that we are just as happy to increase our levels of the design variable (e.g., processor speed) to obtain a corresponding linear gain in the utility, anywhere within this range of interest. Perhaps most important here is the point that, qualitatively, we are simply saying that the straight line defines a range within which we expect to achieve design gains by tuning the design variable in question.

Consider another example. Suppose we are asked to design a Braille printer that is quiet enough to be used in office settings. None of the competing designs are quiet enough to be so used. How quiet does this design really have to be? To answer this question, we must determine the relevant units of noise measurement and the range of values of these units that are of interest. We would also find out how much noise is generated by current printer designs and whether or not listeners can distinguish among different designs. If one printer produces the same noise level made by a pin dropping on a carpet, while another generates the noise level of a ticking watch, we would likely view both as quiet enough to be fully acceptable. Similarly, if one printer is as loud as a gas lawn mower, and another is as loud as an unmuffled truck, there is no utility gained by distinguishing between these two designs as neither design will be used in an office setting. (Note that this example shows a *reverse* S-curve in which we start at saturation because there is no gain to be made at such low levels of quietness, and then we degrade to a level of no utility for printers that are uniformly too loud.)

Since sound intensity levels are usually measured in decibels (dB), we might conclude that some range of dB is likely to be of interest. Carrying this further, we might look

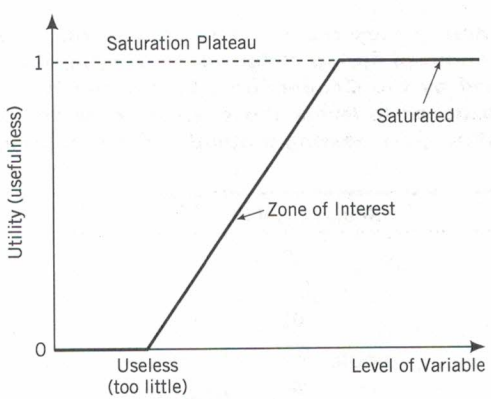


FIGURE 4.4 A linear approximation of the hypothetical performance specification curve shown in Figure 4.3. In this case, the design team has agreed on the lower (minimum) and upper (saturation) levels of interest, and is assuming that an equal increase anywhere along the sloping line brings the user an equal gain.

TABLE 4.1 Sound intensity levels that are produced by various devices and are measured in various environments. Sound intensity levels are measured in decibels (dB) and are a logarithmic expression of the square of acoustic power. Thus, a 3 dB shift corresponds to a doubling of the energy produced by the source, while the human ear cannot distinguish between levels that differ by only 1 dB (or less). (Adapted from Glover, 1993.)

Level (dB)	Qualitative Description	Source/Environment
10	Very Faint	Hearing Threshold; Anechoic Chamber
20	Very Faint	Whisper; Empty Theater
30	Faint	Quiet Conversation
40	Faint	Normal Private Office
50	Moderate	Normal Office Background Noise
60	Moderate	Normal Private Conversation
70	Loud	Radio; Normal Street Noise
80	Loud	Electric Razor; Noisy Office
90	Very Loud	Band; Unmuffled Truck
100	Very Loud	Lawn Mower (Gas); Boiler Factory

for some indication of how much noise is produced by other devices and within different environments. We show sound intensities for various devices and environments in Table 4.1. For reference, we show the noise exposure levels to which workers may be exposed in Table 4.2. These levels, expressed in hours of exposure, are defined by OSHA, the federal agency concerned with the safety of work environments. With such environmental and exposure information in hand, the designer can identify a range of interest for a performance specification for the Braille printer. New printer designs must generate less than 60 dB noise in an office environment. Further, smaller values of generated noise levels are considered gains, down to a level of 20 dB. All designs that generate less than 20 dB are equally good. All design that produce more than 60 dB are unacceptable. Note that any realistic designs will generate noise levels that are so far below the OSHA exposure values that occupational safety is not an issue here.

TABLE 4.2 Permissible noise exposures in American work environments, expressed in intensity levels (dB) permitted during various daily durations (hours). These levels and durations are defined by the Occupational Safety and Health Administration (OSHA), If workers are exposed to levels above these or for period of times longer than indicated, they must be given hearing-protection devices. (Adapted from Glover, 1993.)

Daily Duration (Hours)	Sound Level (dB)
0.5	110
1	105
2	100
3	97
4	95
8	90

4.2.2 Setting performance levels

Performance specifications require sound engineering, reasonable measurements, and clarified client interests.

We now extend the above discussion to setting performance levels. First, we determine design parameters that reflect the functions or attributes that must be measured and the units in which those parameters are to be measured. Then we establish the range of interest for each design parameter. For desirable design variables (i.e., qualities or attributes), utility values *below a threshold* are treated as equals because no meaningful gains can be made. Utility values above a *saturation plateau* are also indistinguishable because no useful gains can be achieved. (We are assuming a standard S-curve in which the threshold comes first and the plateau last.) The range of interest lies between the threshold and the plateau. It is within this zone that the design gains should be matched to and measured with respect to the design parameters that are the subject of a given performance specification. This process works well when we exercise judgment in setting performance specifications based on: sound engineering principles, an understanding of what can and cannot be reasonably measured, and an accurate reflection of both client's and users' interests.

Consider once again the beverage container. Each of the functions that were specified in Section 4.1.2.1 has a range of values that must be specified. Some of those functions and some relevant questions associated with each function are:

- *Contain liquid*: How much liquid must the container contain and at what temperatures? Is there a range of fluid amounts that we can put into a container and still meet our objectives?
- *Resist forces induced by temperature extremes*: What temperature ranges are relevant? How might we measure the forces created by thermal stresses on the container designs?
- *Resist forces induced by handling in transit*: What are the range of forces that a container might be subject to during routine handling? To what degree should these forces be resisted in order for the container to be acceptable?

Note that similar but distinct problems arise for the second and third functions on this list, as they both relate to forces.

We can now develop a set of performance specifications that the container designs should meet by addressing these and similar questions. For example, we might indicate that each container must hold 12 ± 0.01 oz. In this case the specification has become a constraint because the corresponding utility plot is a simple binary switch: We either meet this design specification or we don't. (Of course, it is possible to study the container design problem as one in which a variable single-size serving is possible, in which case there may exist a linearized S-curve for the container size where smaller is better.) Still another performance requirement could emerge from a production concern, namely, that the containers can be filled by machines at a rate of 60–120 containers per minute. Thus, any container that cannot be filled at least this fast creates a production problem, while a faster rate might exceed current demand projections.

We might also specify that the designs should allow the filled containers to remain undamaged over temperatures from –20 to +140°F. Temperatures lower than a threshold of –20°F are unlikely to be encountered in normal shipping, while temperatures higher than a plateau of +140°F indicate a storage problem. It may be that some designs that appeal in other ways are limited here by either temperature extreme. A judgment will have to be made about the importance of this function and its associated performance requirement.

It is also worth noting that the specification of the performance of a device is often published *after* it has been designed and manufactured because users and consumers want to know whether the product is appropriate for *their* intended use. End users, however, are usually not parties to the design process, and so they depend on published performance specifications that set out the performance levels that can be expected from a device or system. In fact, in many instances designers examine the performance requirements of similar or competing designs to gain insight into issues that may affect end users.

4.2.3 Interface performance specifications

As previously noted, performance specifications also specify how devices or systems must work together with other systems. Such requirements, called *interface performance specifications*, are particularly important in cases where several teams of designers are working, on different parts of a final product, and all of the parts are required to work together smoothly. For example, a designer must ensure that the final design of a car radio is compatible with the space, available power, and the wiring harness of the car. Thus, a design team that has divided a project into several parts must ensure that the final parts will work together. In such cases, the boundaries between the subsystems must be clearly defined, and anything that crosses the boundaries must be specified in sufficient detail to allow all teams to proceed.

Interface performance specifications are increasingly important for large firms that, in an increasingly competitive international arena, are trying to minimize the total time needed to design, test, build, and bring to market new products. Most of the world's major automobile companies, for example, have reduced their design and development times for new cars to one-half or less of what they were a decade ago by having design teams work *concurrently*, or at the same time, on many systems or products, all of which must work together and be suitable for manufacture. This puts a premium on the ability to understand and work with interface performance specifications.

Developing interface performance specifications is easy theoretically, but is extremely hard in practice. During conceptual design, the boundaries or interfaces between the systems that must work together must be specified, and then specifications for each item that crosses a boundary must be developed. These specifications might be a range of values (e.g., 5 V, $\pm$ 2 V), or logical or physical devices that enable the boundary crossing (e.g., pinouts, physical connectors), or simply an agreement that a boundary cannot be breached (e.g., between heating systems and fuel systems). In every case the designers of systems on both sides of a boundary must have reached a clear agreement about where the boundary is and how it is to be crossed, if at all. This part of the process can be difficult and demanding in practice since teams on all sides are, in effect, placing constraints on all of the others. A black box functional analysis could be helpful in developing interface specifications because it allows all of the parties to identify the inputs and outputs that must be matched and to deal with any side effects or undesired outputs.

4.2.4 On metrics and performance specifications

The distinction between metrics (defined in Section 3.4) and performance specifications is often a source of confusion. Both involve quantifying or otherwise specifying how well a design does something. In some cases, metrics that are adopted for an objective may also be

used as the specifications for some functions. Nevertheless, there are important differences to keep in mind:

- *Metrics apply to objectives (only).* They allow both designers and clients to assess the extent to which an objective has been realized by a particular design.
- *Performance specifications are typically applied to functions.* They specify how well those functions must be realized by a design.

Metrics measure objectives; performance specs are measures of functions.

Performance specifications also take on something of the characteristics of constraints since any design that fails to meet a performance requirement can be considered to have failed. Metrics are needed for *all* of the objectives that are being considered in the design selection process; performance specifications apply only to functions for which there are definable limits of acceptability.

4.3 FUNCTIONS FOR THE XELA-AID CHICKEN COOP DESIGN

In Chapter 3 we clarified and documented the objectives for the Xela-Aid chicken coop design project and refined the problem statement. Here we present some functional analyses and performance specifications that the student teams developed. Remember that we are presenting student design work to demonstrate the largely successful application of the principles and techniques we are describing. The ability to review design documents is another important design skill that can be honed by reading and evaluating the design efforts of other terms, albeit with patience and prudence.

A chicken coop performs a number of functions. One design team used the enumeration method to develop the list of functions shown in Table 4.3. Notice that many of the functions are of the form "allow...." While this is a reasonable way to begin functional

TABLE 4.3 One list of functions for the Xela-Aid chicken coop, as developed by one of the freshman design teams.

Protect chickens from predators during the day
Protect chickens from predators during the night
Protect chickens from weather
Allow for feeding chickens
Allow for watering chickens
Keep water fresh
Allow for egg collection
Protect eggs
Allow incubation of eggs
Ventilate coop
Allow for nest cleaning
Allow for chicken entry
Allow for waste removal
Allow human entry/exit from coop
Allow human entry/exit from perimeter