Normative Approaches to Work Analysis: "The One Best Way?" 3

> Each employee of your establishment should receive every day clear-cut, definite instructions as to just what he is to do and how he is to do it, and these instructions should be exactly carried out, whether they are right or wrong.
>
> —Frederick Winslow Taylor (cited in Kanigel, 1997, p. 377)

> How is it possible to write a procedure for absolutely every possible situation, especially in a world filled with unexpected events? Answer: it's impossible. [Yet] procedures and rule books dominate industry.
>
> —Norman (1998, p. 156)

PURPOSE

In chapter 2, we argued that it is important to conduct some form of work analysis, and that such an analysis should be based on an ecological approach. As we pointed out in chapter 1, various types of work analysis techniques have been developed since the beginning of this century. In this chapter and the next, we critically examine some of these techniques to see how well they stand up to the challenges imposed by complex sociotechnical systems (see chap. 1). Normative approaches to work analysis, particularly task analysis, are examined in this chapter. To anticipate, we conclude that some form of task analysis is indispensible in complex sociotechnical systems, but that a work domain analysis is also needed to overcome the limitations of task analysis.[1]

THREE GENERATIONS OF WORK ANALYSIS TECHNIQUES

Because many different types of work analysis techniques have been proposed, it is useful to categorize them in a way that highlights important similarities and significant differences as well. Rasmussen (1997a) distinguished between three generic categories of models that can be adopted for the specific purpose of grouping work analysis techniques. *Normative models* prescribe how a system should behave. In contrast, *descriptive models* describe how a system actually behaves in practice. Finally, *formative models* specify the requirements that must be satisfied so that the

[1]As mentioned in chapter 1 (see Footnote 2), task analysis and work domain analysis are two different types of work analysis. One of the main purposes of this chapter is to explain the difference between these two techniques.

system could behave in a new, desired way.[2] Rasmussen used these three model categories to examine the evolution of theories in a number of diverse areas of research, including engineering design, organizational behavior, decision making, and human error. His review revealed a progression over time in each of these areas, from normative first-generation models, to descriptive second-generation models, and only very recently, to formative third-generation models.

The same progression can be seen in the evolution of approaches to work analysis. Seminal Tayloristic work methods analyses (Taylor, 1911), traditional human factors task analyses (R. B. Miller, 1953), and early HCI Goals, Operators, Methods, Selection Rules (GOMS) analyses (Card, Moran, & A. Newell, 1983) are all exemplars of normative approaches to work analysis. Each of these approaches prescribes a normative, rational benchmark for how workers should behave in different situations. More recently, researchers from the anthropological (Hutchins, 1995a; Suchman, 1987), activity theory (Bødker, 1991; Nardi, 1996b), and naturalistic decision-making research communities (Klein, Orasanu, Calderwood, & Zsambok, 1993; Zsambok & Klein, 1997) have all pointed out that workers' actions frequently do not—and indeed, should not—always follow these normative prescriptions. These researchers have come to this conclusion by studying how workers actually behave in situ and by developing descriptive models of current practice. To be clear, the problem with normative models is not that they are normative per se, but that the assumptions they make about human work are not realistic and thus not very useful (Sheridan, in press). But as we show in chapter 4, there are problems in deriving implications for design from descriptive work analysis techniques too, at least for complex sociotechnical systems. These problems have been recognized by researchers from a diverse set of communities, including HCI (Carroll, Kellogg, & Rosson, 1991), sociology (Schmidt, 1991b), anthropology (Button & Dourish, 1996), and naturalistic decision making (T. E. Miller & Woods, 1997). Nevertheless, a solution to these problems has remained elusive.

What we need is a third-generation, formative work analysis framework that helps us specify the design attributes that computer-based information systems should have to satisfy safety, productivity, and health goals.[3] Chapter 5 provides a sketch of the formative approach to work analysis that is the subject of this book. But first, the limitations and implications of normative approaches to work analysis are described in this chapter. Descriptive approaches are described in chapter 4.

Outline

We begin by describing one type of normative approach to work analysis—task analysis. We then show that task analysis techniques actually fall into two qualitatively different categories, constraint and instruction based, with instruction-based techniques being far more prevalent. Our review reveals that instruction-based ap-

[2]Rasmussen (1997a) used the term *predictive* rather than *formative* for this construct. We changed the label because the term *predictive* has connotations that could easily lead to misunderstandings (see Footnote 3).

[3]Note that the requirements that emerge from a formative work analysis do not uniquely specify a new design. That is, they rule out many design alternatives, but additional insight (e.g., human factors and HCI design principles) are required to flesh out a detailed design. This is one of the reasons why Rasmussen's (1997a) label *predictive* can be misleading (see Footnote 2). Formative work analyses are predictive in the qualitative sense used in dynamical systems theory (Port & van Gelder, 1995) but not in the formal, quantitative sense that most of us usually associate with the concept of prediction.

proaches to task analysis are limited in their ability to identify comprehensively the information requirements in complex sociotechnical systems. Fortunately, however, constraint-based task analyses provide a viable alternative that offers some important insights. Their weakness is that they are not well suited to the demands associated with unanticipated events. Work domain analyses are proposed as a complementary technique that can help analysts identify the information support workers need to deal with unanticipated events.

THE NORMATIVE APPROACH: TASK ANALYSIS

Scope

As we mentioned, there are different exemplars of normative approaches to work analysis (e.g., Tayloristic work methods, task analysis, GOMS). Rather than provide an exhaustive and detailed review of each of these individual techniques, we use task analysis as a prototypical exemplar with which to examine critically normative approaches to work analysis.[4]

In this section, we first describe various types of task analysis techniques and provide simple examples of each type. Second, we show how these techniques fall into two qualitatively distinct categories. Third, several important implications for complex sociotechnical systems arising from this distinction are discussed. To anticipate, we show that there is a conflict between constraint- and instruction-based task analysis techniques. This conflict is unraveled in the next section of this chapter.

What Is Task Analysis?

Even the more specific category of task analysis actually represents a diverse set of approaches. In what is perhaps the most comprehensive and clearly-written guide to task analysis produced to date, Kirwan and Ainsworth (1992) reviewed many, if not all, of the major techniques for performing a task analysis. These methods can be used to identify and examine "the tasks that *must* be performed by users when they interact with systems" (Kirwan & Ainsworth, 1992, p. vii, emphasis added). Or in slightly different terms, "Task analysis can be defined as the study of what an operator (or team of operators) is *required* to do, in terms of actions and/or cognitive processes to achieve a system goal" (Kirwan & Ainsworth, 1992, p. 1, emphasis added). These two definitions unambiguously put task analysis techniques into the class of normative approaches to work analysis. The emphasis is on identifying what workers should be doing to get the job done, reminiscent of Taylor's quest for the "one best way" (Kanigel, 1997). Our story in the Preface provides an example: The regulators were looking for "procedural compliance" when evaluating the nuclear power plant operators' performance in the simulator. Despite their many differences, task analysis techniques generally share this rational quest for identifying the ideal way(s) in which the job should be performed (e.g., Kirwan, 1992).

[4]Two caveats are in order. First, we use the term *task analysis* to refer to the end product (i.e., the model or representation) generated by the process of studying a task. This is a typical convention in North America, although in some cases the term *task description* is used for this purpose instead (Singleton, 1974). Second, task analysis can be used in either a descriptive or a normative fashion, but we only focus on the latter in this chapter. Descriptive forms of work analysis are the subject of chapter 4.

To be sure, the precision with which this objective is sought differs considerably among task analysis techniques. Three different levels can be identified (for examples of task analysis techniques at these levels, see Kirwan & Ainsworth, 1992):

1. *Input–Output*—This first level identifies the inputs that are required to perform the task, the outputs that are achieved after it is completed, and the constraints that must be taken into account in selecting the actions that are required to achieve the task. An example is illustrated in Fig. 3.1 for the task of determining the rate of gasoline consumption in km/liter for an automobile. The two inputs and the single output for the task are shown, but the steps that are used to achieve the task are not specified. Note that the inputs are specified in miles and gallons, so a conversion process is required to perform the task correctly. Thus, certain constraints must be obeyed to reach the goal state. For the example in Fig. 3.1, two relevant constraints are the number of kilometers in a mile and the number of liters in a gallon. Even for a task as simple as this one, there are several different sets of actions that we could adopt to perform the same task. For example, we could use a calculator, in which case the main steps are reading off numbers from the odometer and the gas pump and then typing those numbers into the calculator. Or alternatively, we could use mental arithmetic, in which case some cognitive steps are also required to arrive at the correct answer. The input–output level deliberately ignores these process details. It only provides a very high level product description of the task.

There are several points worth noting here. First, the constraints on action can be of different types. In the example presented in Fig. 3.1, the constraints were relationships between variables. However, they could also be sequential in nature (e.g., there must be water in the kettle before you turn the stove on). Second, it is not possible to perform the task correctly without factoring in these constraints. They must be taken into account, independent of how the task is actually performed. Third, the constraints serve the function of "problem state space reduction" (Sheridan, in press) but they usually do not specify a unique action flow sequence. They merely limit the options (degrees of freedom) on viable action sequences (i.e., all sequences that do not factor in the correct conversion factors are ruled out). In the words of Kugler, Kelso, and Turvey (1980), "it is not that actions are caused by constraints, it is rather that some actions are excluded by them" (p. 9).

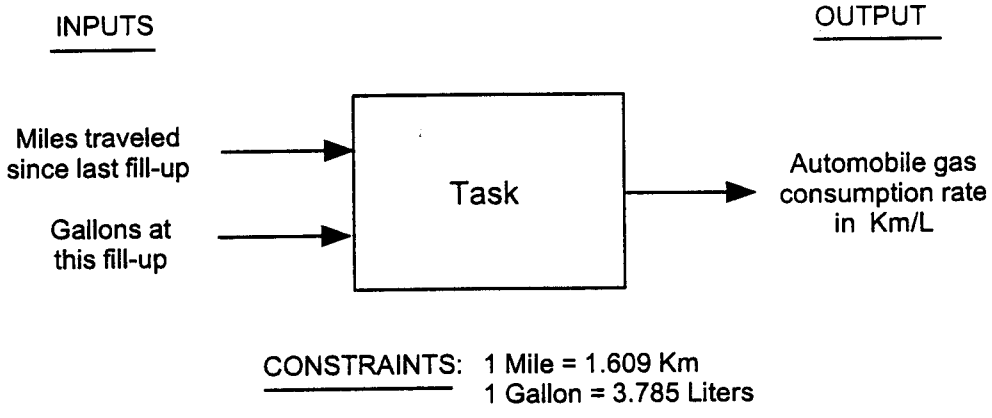

FIG. 3.1. Example of input/output task analysis for determining the rate of gasoline consumption in an automobile.

2. *Sequential Flow*—This level identifies a temporally ordered sequence of actions that is required to complete the task. Such a description is equivalent to a flowchart of the process that workers should follow to perform the task, much like the specification of a computer algorithm. Traditionally, the steps in the flow sequence have been overt actions, although cognitive steps can also be specified. The steps themselves are usually taken from a behavior taxonomy such as the classic Berliner taxonomy, shown in Table 3.1. The column on the extreme right provides a "periodic table" of elemental action verbs that can be used to compose a flow sequence for a particular task. For example, one specific sequence of actions for performing part of the task in Fig. 3.1 could be: (a) *Read* current odometer value, (b) *read* odometer value at last fill-up, and (c) *calculate* the difference to obtain miles traveled since last fill-up. The result would be an algorithm for performing the task in a particular way. Note that, because it is more detailed, this level of task analysis is usually dependent on the device workers currently have available to perform the task. For example, if you had a trip odometer in your car, the action sequence just specified would be unnecessary, as long as you reset the odometer the last time you filled your tank. Instead of calculating the miles traveled since the last fill-up, you could simply read off this value from the trip odometer.

Like computer program flowcharts, the flow sequence can include branch points to represent alternative paths in the process sequence (Meister, 1985). In practice, however, this branching is avoided to the extent possible because it makes for a more complicated task representation (Meister, 1995a). Another example of a task analysis conducted at the sequential flow level is shown in Fig. 3.2. Note the single sequence of overt actions.

3. *Timeline*—This final level identifies a temporally ordered sequence of actions that is required to achieve the task, with duration estimates for each action. This level is the most detailed of all, and was traditionally used in forced assembly line operations. Sticking with the example we used in the previous level, we might obtain the following task description (the time estimates are purely hypothetical): (a) 0–1 s: *Read* current odometer value, (b) 1–2 s: *read* odometer value at last fill-up, and (c) 2–3 s: *calculate* the difference to obtain miles traveled since last fill-up. This type of task analysis should be familiar to industrial engineers who have been exposed to time and motion studies (Maynard, 1971), to HCI professionals who have been exposed to GOMS and the Model Human Processor (Card et al., 1983), and to human factors professionals who have been exposed to time-based measures of mental work load (Meister, 1985). Not only is the sequence of actions that should be performed specified, but so is the duration of each of those actions. A more complex example is illustrated in Fig. 3.3, involving time sharing of multiple tasks shown by the overlap in the timeline. If we are to believe this analysis, there is only one right way to perform this task at this level of resolution. All of the discretion has essentially been eliminated, leaving only a single, very well defined procedure for workers to follow.

Constraints Versus Instructions

These three levels of task analysis, summarized graphically in Fig. 3.4, differ in their specificity. If we adopt a multidimensional state space as a frame of reference, the first level of task analysis (input–output) shown in Fig. 3.4a merely identifies a *point* in the

TABLE 3.1 Classification of Task Behaviors

Perceptual processess	Searching for and receiving information	Detects Inspects Observes Reads Receives Scans Surveys
	Identifying objects, actions, events	Discriminates Identifies Locates
Mediational processes	Information processing	Categorizes Calculates Codes Computes Interpolates Itemizes Tabulates Translates
	Problem solving and decision-making	Analyzes Calculates Chooses Compares Computes Estimates Plans
Communication processes		Advises Answers Communicates Directs Indicates Informs Instructs Requests Transmits
Motor processes	Simple/Discrete	Activates Closes Connects Disconnects Joins Moves Presses Sets
	Complex/Continuous	Adjusts Aligns Regulates Synchronizes Tracks

Note. From Meister (1985). Copyright 1985 by John Wiley & Sons, Inc. Reprinted by permission.

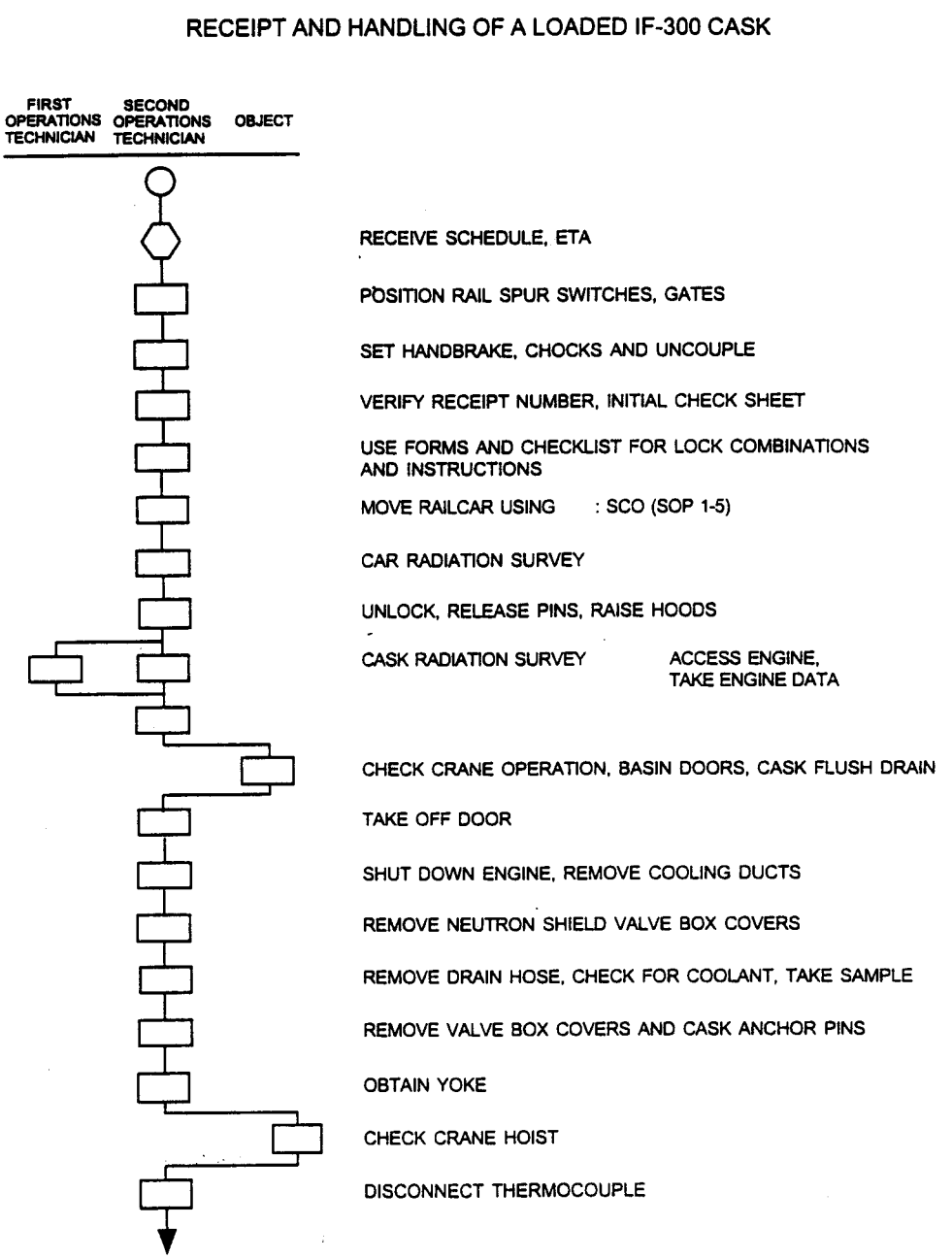

FIG. 3.2. Example of sequential flow task analysis for the receipt and handling of a loaded cask.

space (i.e., the goal state), the inputs that are required to get to that point (not shown in the figure), and the constraints that must be respected (i.e., the areas that must be avoided) if workers are to get to the goal state. The second level of task analysis (flow sequence), shown in Fig. 3.4b, represents a marked increase in specificity. Not only is the goal state identified, but so is a sequence of actions for getting to the goal state. There is still some leeway, however, because the timing of the actions is not specified. Therefore, in terms of a state space representation, this level of task analysis corre-

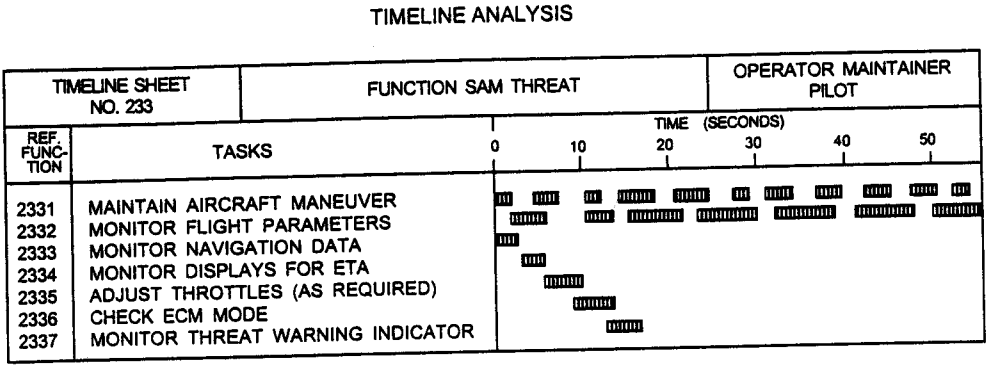

FIG. 3.3. Example of a timeline task analysis for a military aviation task (Meister, 1985). From *Behavioral analysis and measurement methods* by David Meister, Copyright © (1985, John Wiley & Sons, Inc). Reprinted by permission of John Wiley & Sons, Inc.

sponds to a *path* that must be followed (see Fig. 3.4b). Finally, the third level of task analysis (timeline) is the most specific of all. It prescribes a sequence of actions, each with a particular duration. This level of analysis can be represented as a unique *trajectory* in the state space for reaching the goal (see Fig. 3.4c).[5]

The graphical comparison in Fig. 3.4 clearly reveals an otherwise subtle qualitative distinction between the three levels of task analysis. The first level, being based on *constraints,* specifies what should not be done, whereas the next two levels, being based on *instructions,* specify what should be done (cf. Kugler et al., 1980). This qualitative difference, although perhaps unfamiliar to us, is of absolutely fundamental importance to the approach taken in this book. Therefore, it is discussed at length in chapter 5 and revisited several times in later chapters.

It is important to add that instruction-based techniques are, by far, the more prevalent of the two types of task analysis. For example, almost all of the techniques described in Kirwan and Ainsworth's (1992) guide to task analysis are instruction based. This observation has important implications that are highlighted later in this chapter.

Implications

The point of doing any kind of work analysis is to derive implications for systems design. Different forms of work analysis make different assumptions about the nature of work, so they lead to different designs, which, in turn, lead to different types of guidance for workers. For example, constraint-based and instruction-based approaches to task analysis have different implications for workers. First, instruction-based approaches are more detailed, so they leave less discretion to workers (cf. Leplat, 1989). At the lowest level (timeline, see Fig. 3.4c), the analyst must make decisions about what actions should be performed to accomplish the task, how they should be sequenced together, and how long each will take. There is very little left

[5]Note that the distinction between a path and a trajectory is a relative one. For example, if we adopt a more fine-grained level of analysis (i.e., if we "zoom in"), what looked like a trajectory becomes a path. Conversely, if we adopt a more coarse level of analysis (i.e., if we "zoom out"), what looked like a path can look like a trajectory.

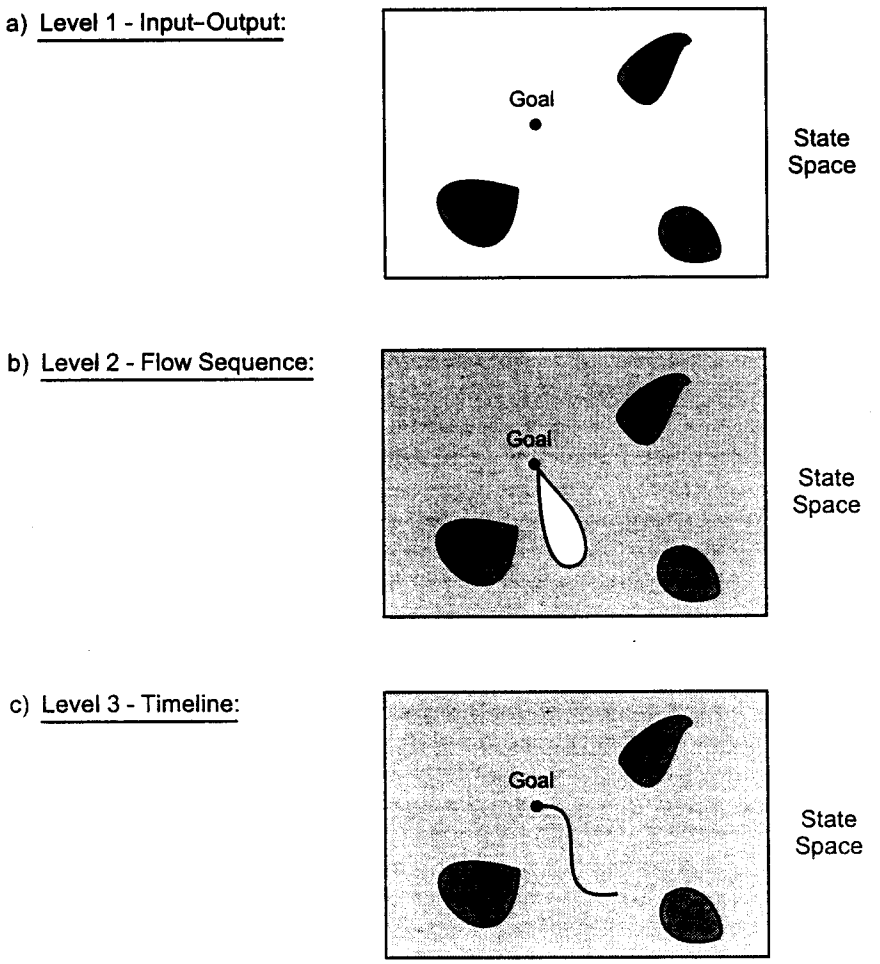

FIG. 3.4. A qualitative graphical comparison of the specificity associated with the three levels of task analysis described in the text. Shaded regions represent hypothetical areas that should be avoided, according to the analysis.

for the worker to do except follow the resulting procedure or work flow, reliably and consistently (Reed, 1996). With constraint-based approaches, on the other hand, none of these decisions are made up front by the analyst. Only guidance about the goal state and the constraints on action are provided, not about how the task should be accomplished (Fig. 3.4a). Consequently, workers are given some discretion to make decisions online about how to perform the task. This contrast shows that constraint-based approaches to task analysis provide more worker discretion than do instruction-based approaches. And as we mentioned in chapter 1, decision latitude is a key factor in determining worker health.

Of course, we could just as easily turn these observations around by examining, not the discretion, but the guidance offered to workers by the two types of task analysis. We could argue that instruction-based approaches provide more detailed guidance and thus are less likely to lead to human error. In contrast, constraint-based approaches offer minimal guidance and thus are more likely to lead to human error. Thus, whether instruction-based or constraint-based approaches to task analysis are

better depends on the view that we hold of people (cf. Reed, 1996; Sachs, 1995). If we view people in an optimistic light, we may want to adopt the constraint-based approach—provide a computer system with information about the goal to be achieved, the current state of affairs, and the constraints to be obeyed, and then let workers decide how to achieve the goal. If we view people in a pessimistic light, then we may want to adopt the instruction-based approach—identify up front the "correct" way to perform the task, and then build the results of that analysis into a computer-based work flow that leads workers along and enforces that "proper" way of doing the job (e.g., Andognini, 1997; Marsden, 1996).

Second, this difference between levels of analysis can also be viewed from the perspective of accommodating performance variability (cf. T. J. Smith, Henning, & K. U. Smith, 1994). As shown in Fig. 3.4, constraint-based analyses accommodate a great deal of variability in worker action (i.e., many ways of doing to the task), whereas instruction-based analyses accommodate very little variability in worker action (i.e., in the extreme, only one way of doing the task). This observation has implications for two very important factors, namely the extent to which workers encounter situations that foster learning and the ability that workers are given to deal with unusual circumstances.

With regard to opportunities for learning, in the subarea of control theory known as system identification (e.g., Johannson, 1993), it is well established that it is necessary to "stimulate" a system by varying its control inputs widely to obtain a comprehensive understanding of the characteristics of that system. Also, in the subarea of control theory known as adaptive control (e.g., Narendra, 1986), it is well known that action variability provides the "persistent excitation" that is required for adaptive control. Furthermore, in psychology it is well established that providing opportunities for people to engage in exploratory behavior fosters learning (E. J. Gibson, 1991). In motor control, researchers have found that the inherent variability in the human motor system serves an educational function because it allows people to experience the varying set of conditions that are required for improved motor skill (Bernstein, 1996). These converging findings from disparate research areas show that there is a very robust relationship between action variability and learning opportunities (cf. Norros, 1996).

As for the ability to deal with novelty, if the conditions under which a task is to be performed vary from one situation to the next (e.g., starting to the right of the goal in Fig. 3.4 rather than to the left), then the actions that are required to get to the goal must differ for the two cases (cf. Bartlett, 1958; Bernstein, 1996). This relationship is explored in detail in the next section. For now, we merely observe that constraint-based approaches to task analysis provide workers with more flexibility and better opportunities to learn than do instruction-based approaches.

Third, recall that the definitions provided earlier indicate that task analysis identifies the tasks that *workers* are required to perform. But, at a detailed level, what workers are required to do depends on the device they have available to them. Note that, in our terms (see Glossary), *device* refers to the interface and automation but not to the work domain. Thus, any detailed account of a task is likely to make assumptions about the device that workers have available to perform the task. Instruction-based analyses are subject to this limitation. We saw this in the very simple case of figuring out the gas consumption for an automobile. If our car has a trip odometer, then the actions that we must perform to complete the task are different from those that are required if a trip odometer is not available. Similarly, if a calculator is available, the

set of actions that we need to perform to complete the task will also change, in part because some steps have been automated and are thus not under workers' purview. Not only that, but even the type of calculator makes a difference. If automatic conversion functions are available, then we do not have to know what the multipliers are; otherwise we do. Therefore, instruction-based analyses are more *device-dependent* (Benyon, 1992a) because their content and form changes as a function of the interface and automation that workers have available to perform the task.

In contrast, because constraint-based analyses merely provide an envelope on task achievement, they are more *device-independent.* For example, the requirements shown in Fig. 3.1 must be taken into account if the task is to be accomplished correctly and consistently. If the fact that there are 3.785 liters in a gallon is not taken into account somehow, whether it be by the worker or by the device (e.g., a calculator), then the correct answer cannot possibly be consistently obtained. It is worthwhile emphasizing that this statement is true, even if the actor performing the task is not a person. An ant, for instance, would have to deal with the same constraints. In general terms, some relationships are properties of a task and quite independent of any particular device (Diaper & Addison, 1992). By constraining rather than instructing workers, constraint-based analyses describe properties of the task, rather than properties of how to do the task with a particular device. This difference may be subtle, but it is quite important. Of course, the device-dependent/independent distinction lies on a continuum rather than in the two discrete, bipolar categories we have been describing, but this does not change the main point of our argument. The level of detail at which a task analysis is conducted is proportional to the device-dependence of that analysis. Therefore, constraint-based analyses are generally less device-dependent than are instruction-based analyses.

Consequently, instruction-based analyses are plagued by a fundamental logical problem that has very important design implications. If an instruction-based task analysis is going to be performed, then assumptions about the design of the device (i.e., computer interface and automation) available to workers must be made. The reason for this is that the particulars of the timeline or flow sequence will change as a function of the device that is available (see earlier discussion). But, the reason we want to do a task analysis in the first place is to figure out how the device should be designed. We are thus faced with an infinite regress that is apparent, albeit implicit, in the practical guide to task analysis by Kirwan and Ainsworth (1992):

- Task analysis defines what a *worker* is required to do.
- Task analysis can be used by designers at an early stage of design to decide how to allocate functions between workers and machines.
- Some tasks are better performed by workers whereas other tasks are better performed by machines.

What is left implicit is the fact that different function allocation policies lead to different worker responsibilities. For instance, if a task is completely automated, then there may be no worker activities for that task. At the other extreme, if there is no automation, then all of the activities may have to be performed by the worker. The very same logic applies to interface design. Returning to the trip odometer example, with one interface a worker may have to perform one action (calculate the difference between current and previous km), whereas with a different interface a worker may have to perform a

very different action (read the trip odometer) to achieve *the very same task goal.* So we are left with a conundrum where "what the worker is required to do" is both an input to, and an output of, the design process. Note that it is not possible to escape this dilemma completely by iteration because any design decision can result in a change in worker activities or responsibilities. The implications of this task-artifact cycle (Carroll et al., 1991) are discussed in more detail in chapter 4.

Conclusions

What can we conclude from this review of task analysis techniques? The primary advantage of the instruction-based approach seems to be that more guidance is provided. By following the steps laid out in a flow sequence or timeline, workers may be less likely to forget a step, perform the wrong step, or perform the right steps in the wrong order. In short, by providing workers with a "precompiled" solution to task demands, there may be less chance of human errors arising from human information-processing limitations.

The constraint-based approach appears to have several advantages. First, more discretion is given to workers to decide exactly how the task should be performed. Based on Karasek and Theorell's (1990) demand-control model (see chap. 1), we would therefore expect better worker health. Second, constraint-based approaches also accommodate greater variability in action. As a result, workers have more opportunities to learn and better chances of coping with unusual or changing circumstances. Thus, constraint-based approaches foster learning and support greater flexibility. Third, constraint-based approaches make fewer assumptions about the properties of the device (i.e., interface + automation) available to workers. Consequently, it is more likely that the new design will result in new functional possibilities, rather than being constrained by designers' current assumptions about functionality. Given Landauer's (1995) arguments and evidence (see chap. 1), we would thereby expect better productivity.

When taken together, these arguments seem to point in two conflicting directions. On the one hand, it seems that we should adopt instruction-based approaches to task analysis to provide workers with detailed guidance that they can use to perform tasks with minimal errors. On the other hand, it seems that we should adopt constraint-based approaches to give workers more discretion, accommodate more variability in action, and identify new possibilities for doing the job, thereby resulting in better health, flexibility, and productivity, respectively. Given that instruction-based approaches are, by far, the most prevalent, we might think that their benefits outweigh those of constraint-based techniques. This possibility is explored next.

CONSTRAINTS OR INSTRUCTIONS? THE VIEW FROM CONTROL THEORY

Control theory provides a generic language for understanding action (Flach, 1990a; Powers, 1973a, 1973b; T. J. Smith et al., 1994), and so we use it here to resolve the apparent conflict between constraint-based and instruction-based approaches to task analysis. Although control theorists have developed sophisticated mathematical formalisms for analyzing and representing systems, we only use qualitative concepts

from control theory in an informal way to simplify the discussion. In this subsection (based on the work of Marken, 1986; Powers, 1973a, 1973b, 1978), we present two models of goal-oriented behavior, the second being more complex than the first. The basic assumption behind each model is that people are adaptive, goal-oriented agents. Although they are not realistic representations of human action in complex systems, the models can nevertheless be used to uncover a number of important insights.

Simple Negative Feedback Loop

Figure 3.5 shows the simplest possible model of goal-oriented behavior, a negative feedback loop. The variables in the loop are:

Goal *(g)*—the desired state to be achieved and maintained.

Output *(o)*—the output of the system (i.e., the current state of the plant).

Error *(e)*—the difference between the desired state and the current state.

Action *(a)*—the action of the worker.

Although our notation does not make this explicit, each of these variables is dynamic. In addition to these four variables, the negative feedback loop is also characterized by two parameters:

W—the control strategy used by the worker.

P—the dynamics of the plant (the control-theoretic term for the system being controlled).

Given these definitions, a number of deductions can be made. First, the action is determined by the error signal and the strategy used by the worker:

$$a = \mathrm{W}\ e. \tag{1}$$

The error signal, in turn, is the difference between the goal state and the output state:

$$e = g - o. \tag{2}$$

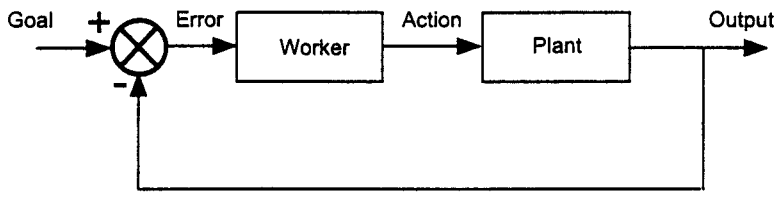

FIG. 3.5. A very simple model of goal-directed behavior.

Substituting Equation 2 into Equation 1 leads to the following:

$$a = W\,(g - o). \tag{3}$$

That is, the actions taken by workers are determined by the goal state, the state of the plant, and the control strategy used by workers. If we know these three terms, then we can predict the sequence of actions that workers should take. This is precisely the approach adopted by instruction-based task analyses (see earlier discussion). The analyst specifies the goal *(g)*, *assumes* a set of initial conditions (*o* at time t = 0) and the strategy that the worker will use (W), and then derives a flow sequence or timeline of actions *(a)* that will satisfy the goal (e.g., Kirwan, 1992).

However, in complex sociotechnical systems, the initial conditions frequently cannot be known with certainty. For example, when a particular task is to be performed on a flexible manufacturing system, the initial configuration of that system may vary from one situation to the next. Similarly, the initial state of a petrochemical plant may also vary over time when performing say a start-up task. What happens in these types of situations? Because *o* at time t = 0 is unknown, *a* cannot be predicted by the analyst. This simple lesson is familiar to anyone who has ever tried to follow a set of instructions for using say a home appliance after having made an error. The instructions (basically a list of actions) are based on certain assumptions about the initial state of the world. When we make a mistake, those assumptions are no longer valid. As a result, we can no longer use the instructions to perform the task, unless we can find a way to resynchronize ourselves with the states that are assumed in the instructions. The very same conclusion applies when conducting an instruction-based task analysis—if the initial conditions are unknown, then we cannot predict the actions that workers need to take to perform the task.

There is a similar problem arising from uncertainty associated with W, the worker's strategy. In complex sociotechnical systems with many degrees of freedom, there can frequently be more than one strategy to achieve the very same task goal (see chap. 9). For example, if we emphasize speed, then one strategy may be viable; whereas if we emphasize accuracy, a different strategy may be preferable. Thus, we may not be able to predict the actual strategy used by workers, particularly if different strategies are used by different workers or by the same worker on different occasions. Under these conditions, *a* will again be unpredictable because of its dependence on W. Thus, there are two reasons why we may not be able to predict the precise actions required to perform a particular task, both causing difficulties for instruction-based task analysis techniques.

We can gain additional insights into the relationship between the predictability of conditions, the variability in action, and the feasibility of constraint- and instruction-based forms of task analysis by adopting a slightly more complex model of goal-directed behavior.

Negative Feedback Loop With a Disturbance

Figure 3.6 illustrates a negative feedback loop like the one in Fig. 3.5, except that a source of disturbances *(d)* has been added to the output of the plant. Disturbances are factors or events that affect the state of the plant in ways that have not been, or cannot be, anticipated by system designers. A very simple example is the wind that can push your car when you are driving. Although designers can expect that wind

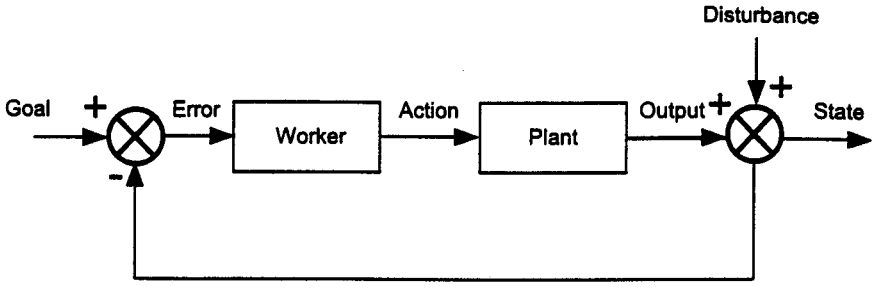

FIG. 3.6. A slightly more complex model of goal-directed behavior.

gusts will occur, they cannot anticipate their magnitude and timing. The wind gusts *(d)* are added to the output *(o)* of your car and thereby cause a different state of the world *(s)* (i.e., the car is in a different position than it would have been in without the wind). In the simpler previous model (Fig. 3.5), the output *(o)* and the state of the world *(s)* were the same thing because there was no disturbance *(d)*. However, in this more complex model, the output *(o)* and the state of the world *(s)* are different because of the presence of the disturbance *(d)*. Many kinds of disturbances can be found in complex sociotechnical systems, some relatively minor and some more complex: an unanticipated change in the economy in the financial industry, a rush order in the manufacturing industry, an unanticipated network failure in the telecommunications industry, an unexpected allergic reaction of a patient to an antibiotic in the medical domain, and multiple, interacting equipment failures in a nuclear power plant (cf. Norros, 1996). The severity of these examples should make it clear that it is important to understand the implications of disturbances for the control of complex sociotechnical systems. Although there can be many different sources of disturbance, for now, we do not discriminate between them. All of them are lumped together in the variable *d*.

As with the previous case, a number of deductions can be made from this very simple model. The actions are still a function of the error signal and the strategy used by the worker:

$$a = \mathrm{W}\ e. \tag{4}$$

In this case, the error signal is the difference between the goal state and the state of the world:

$$e = g - s. \tag{5}$$

The output of the plant is determined by the actions performed by the worker and the dynamics of the plant itself:

$$o = \mathrm{P}\ a. \tag{6}$$

Also, the state of the world is a simple function of two variables, the disturbance and the output from the plant:

$$s = o + d. \tag{7}$$

Substituting Equation 7 into Equation 5 leads to the following:

$$e = g - (o + d). \tag{8}$$

If we substitute Equation 6 into Equation 8, we get:

$$e = g - (\mathrm{P}\ a + d). \tag{9}$$

For the goal to be satisfied, the error signal must go to zero. If we let $e = 0$ in Equation 9, we can see what relationship must be true for the goal to be achieved:

$$g = a\,\mathrm{P} + d. \tag{10}$$

Solving for the actions that the worker must perform, we get:

$$a = (g - d) / \mathrm{P}. \tag{11}$$

Equation 11 has a straightforward, yet very important, qualitative interpretation. The actions that workers must perform *(a)* to achieve the goal are a function of the goal itself *(g)*, the dynamics of the plant (P), and the disturbance acting on the system *(d)*. But as we mentioned before, the disturbance—by definition—cannot be predicted. It follows that the actions that workers need to take to satisfy the goal also cannot be predicted.[6]

The example mentioned earlier of driving a car under windy conditions gives an intuitive appreciation for this fact. If the goal *(g)* is to be satisfied (e.g., stay in your lane), the actions *(a)* performed by the driver must oppose the disturbances *(d)* caused by the wind. If the wind pushes the car to the left, the driver must respond by steering to the right, and vice versa. But because nobody (including the driver) can predict what the pattern of disturbances will be, the precise actions that the driver must take, as well as their timing, cannot be planned up front. Instead, they must be generated online, in real time, as the wind acts on the car.

There are several points that follow from this observation. First, the very same set of actions will have different effects at different times because the pattern of disturbances will rarely be repeated. Each situation is a unique context characterized by a different set of contingencies. Second, as a result of this situated context, if the same goal is to be achieved on different occasions, then a different set of actions will be required to achieve the very same task goal. Thus, the actions have to vary if the outcome is to remain the same. This phenomenon has been referred to as *context-conditioned variability* in the motor control literature (Turvey, Shaw, & Mace, 1978), *unanticipated variability* in the cognitive engineering literature (Roth, Bennett, & Woods, 1987), and *situated action* in the cognitive science literature (Suchman,

[6]Some of you might find it difficult to reconcile the fact that the workers' actions cannot be predicted in these two negative feedback models with the fact that negative feedback control systems work so well in many applications. In fact, there is no conflict because feedback control systems do not predict their actions either. Instead, these are generated online, in real time, as a function of the current context.

1987). We adopt the term context-conditioned variability because it has historical precedence and because it seems to be the most transparent label for this important phenomenon.

Resolution

We are now in a position to try to resolve the apparent conflict between constraint- and instruction-based approaches to task analysis described earlier. The feasibility of the two types of analysis can be understood by distinguishing between closed and open systems. *Closed systems* are completely isolated from their environment. From the viewpoint of the analyst, the behavior of the system can be well understood by examining influences that are internal to the system itself. Conversely, *open systems* are subject to influences (i.e., unpredictable disturbances) that are external to the system.[7] Note that the closed–open system distinction lies on a continuum. Some systems may be more open (i.e., be affected more by disturbances) than others.

With this distinction in hand, the apparent conflict between conducting an instruction-based analysis to reduce human error and a constraint-based analysis to support safety, flexibility, and health may be resolved in a systematic way. The more closed a system is, the more amenable it is to instruction-based forms of task analysis. More concretely, in cases where the analyst can anticipate the conditions under which the work will be done, a realistic and appropriate flow sequence or timeline may be specified. Such conditions are usually obtained in highly proceduralized tasks where the initial conditions are predictable and there are not that many correct ways to perform the task (i.e., the degrees of freedom are few or none). Conversely, the more open a system is, the less amenable it is to an instruction-based form of task analysis. Because there are unpredictable external disturbances acting on the system, it will not be possible to accurately preidentify the different flow sequences or timelines that lead to the satisfaction of the goal. Open systems give rise to context-conditioned variability. Workers must adapt online in real time to disturbances that cannot possibly be foreseen by analysts (Hirschhorn, 1984). Such conditions are usually obtained in cognitive tasks where there are many ways of achieving the same end (e.g., creative problem-solving tasks), or in dynamic systems whose demands change in an unpredictable manner (e.g., Norros, 1996).

Given this insight, we can understand why instruction-based task analysis techniques have been used effectively by system designers for many years, and why they are much more prevalent than constraint-based approaches. In the past, the demands on workers have been comparatively procedural in nature (see chap. 1), the prototypical example being forced assembly line operations. Consequently, analysts could treat such systems as being essentially closed for practical purposes.

---

[7]The system is defined by the analyst (i.e., it is an epistemological entity, not an ontological entity). As a result, if the model adopted by the analyst has gross deficiencies, then the system will be open because it is being affected by factors that are not accounted for in the analyst's model. Therefore, at any one point in time, it is generally not possible to distinguish between the following two situations: (a) External factors are affecting the behavior of the system, and (b) internal factors not accounted for by the analyst's model are governing the behavior of the system. Only when better models are developed can it be established that a situation that looked like case (a) actually turned out to be case (b). For our purposes, it is sufficient to include both cases under the definition of open system because each presents the analyst with sources of disturbance that—by definition—cannot be anticipated.

In these cases, instruction-based analyses were feasible and captured task demands in a way that was sufficiently accurate to derive useful design implications. But more recently, as the demands on workers have become more complex, systems are becoming more and more open (see chap. 1). Specifically, complex sociotechnical systems are subject to external disturbances (e.g., faults, changes in demands) that must be compensated for; they are subject to various forms of uncertainty that must be accommodated; their demands are primarily cognitive and social in nature, so there is no one right way of getting the job done; their behavior is dynamic, requiring workers to adapt to moment-by-moment changes in context. Collectively, these factors create a need for context-conditioned variability that is substantial enough in magnitude that it cannot be meaningfully ignored (cf. Norros, 1996). Our story in the Preface is a vivid indication of this fact. Operators sometimes *had* to deviate from the "one best way" embedded in their procedures, otherwise the desired goal state would not be achieved. As the quote from Norman (1998) at the start of this chapter indicates, the general point is that, in complex sociotechnical systems, it is not possible to identify a flow sequence or timeline that is appropriate for every possible situation. In other words, instruction-based task analysis techniques do not do justice to the richness of the set of actions that are required to cope with the entire set of job demands. The flow sequence and timing of the actions that will satisfy even anticipated goals are not fixed, are too numerous to enumerate, and are subject to factors that cannot be identified a priori (for an example, see Ujita, Kawano, & Yoshimura, 1995).

These limitations have been identified previously, not just by critics of instruction-based task analysis (e.g., Bannon & Bødker, 1991; Greif, 1991; Hirschhorn, 1984), but even more important, by its proponents as well:

- "Workload assessment methods (such as timeline analysis) are relatively crude and measure the operator time involved at a console but not the cognitive demands or 'burden' actually levied on the operator in non-routine situations. More sophisticated technology needs to be developed" (Kirwan, 1992, p. 386).
- "As the cognitive (i.e., mental) content increases, [flow sequence] representations become less satisfactory. In particular there may be many potential internal cognitive mechanisms and the real cognitive structures used by the operators may be 'opportunistic', rather than clear decisions made on strict criteria. Modeling such activity can be difficult" (Kirwan & Ainsworth, 1992, p. 94).
- "As with timeline analysis, [task analytic] simulation techniques are currently poor at covering mental aspects of workload" (McLeod & Sherwood-Jones, 1992, p. 308).
- "Decision making and problem solving . . . activities can be solved successfully by several different strategies and the individual choice will depend on very subjective criteria" (Meister, 1995a, p. 120).

Although these limitations are frequently ignored in practice (e.g., see some of the case studies in Kirwan & Ainsworth, 1992), there seems to be little doubt that instruction-based forms of task analysis are strongly limited for open systems, such as those described in chapter 1, because they tend to ignore the variability and

richness of action in such systems. This limitation becomes especially significant when we consider that the vast majority of existing task analysis techniques are instruction based, and thus not particularly well suited for our purposes.[8]

This criticism is not a call to completely abandon procedures. On the contrary, procedural guidance of some type has a useful role to play in complex sociotechnical systems (see chap. 8). The point is instead that instruction-based approaches are rigid and that we must find some way of supporting the context-conditioned variability needed for open systems.

Who Cares? Design Implications

What are the practical implications of this criticism for design? What happens if we design a computer-based aid for an open system using an instruction-based task analysis? Will it be "good enough," or will it lead to significant problems that merit our attention?

Bisantz et al. (1996) provided a cogent example of what happens when an instruction-based form of task analysis is used to design a computer aid to support workers in an open system requiring context-conditioned variability. They conducted a field study of a computer information system that was intended to help cooks in fast-food restaurants make decisions about what type of food to cook, how much to cook, and when to cook it. There are conflicting criteria governing these decisions: (a) the cooks need to have enough food already prepared to meet customer demand in a timely fashion, and (b) the cooks should not prepare too much food that will not be purchased, otherwise it will go to waste.

The restaurants that were observed in the field study had installed an information system that provided minute-by-minute cooking instructions to assist cooks in making the decisions identified above. The prescriptions made by the aid were projections of appropriate actions, based on predicted sales patterns, actual current sales, and the amount of each product currently on hand. Most of these inputs had already been stored in the computer system by its designers, but the amount of each product currently on hand had to be entered in manually by the cooks. In other words, the information system prescribed to workers what actions were to be performed, in what order, and at precisely what time, based on preidentified variables and relationships. We can infer from this that the computer system was based on some type of instruction-based analysis.

An important characteristic of this work environment that was not taken into account in the design of the aid is that there are several sources of disturbances. For example, customer demand is not completely predictable. To be sure, there are certain trends as a function of the time of day (e.g., lunch, dinner). However, unexpected rushes can and do occur due to local community happenings or due to busloads of travelers coming off of the highway. There is also a disturbance (or source of uncertainty) introduced by the fact that cooks would sometimes forget to input the amount of product on hand, enter it incorrectly, or enter it only after a

[8]It is possible, of course, that some tasks in some complex sociotechnical systems may be so constrained that there is only one way to perform the task safely or efficiently. In such rare cases, a constraint-based task analysis would reveal that there are no degrees of freedom for action, and that a detailed procedure would in fact be appropriate (see chap. 5). The result in this case would be similar to that obtained from an instruction-based task analysis.

substantial delay. The amount of product currently available is one of the inputs into the algorithm governing the recommendations made by the aid, so if there are errors, missing data, or old data in the input, the recommendations made by the aid will not be appropriate. This possibility was not taken into account by the rationalized, ideal assumptions built into the design of the computer aid. In summary, there are external disturbances making this an open system.

The control-theoretic analysis presented earlier allows us to predict how well this computer aid should function in such an environment. The aid is essentially trying to enforce a particular way of doing the task that is rational given the assumptions built into the aid. We would thereby expect that the computer system should make it very difficult, or perhaps even impossible, to accomplish the task in any way other than the rationalized set of actions prescribed by the aid. And because this environment is subject to external disturbances, it should be difficult for cooks to use the computer aid and still adapt to the contingencies of the moment. In other words, because the aid seems to have been designed according to an instruction-based task analysis, it should not be flexible enough to support the context-conditioned variability that is required to achieve task goals in this open system.

The empirical observations of Bisantz et al. (1996) are entirely consistent with this prediction. Because the aid was too rigid, it went essentially unused, despite the fact that there was considerable pressure imposed on workers by management to obey the aid's cooking instructions. Note that this finding was obtained in a comparatively simple fast-food restaurant, not a nuclear power plant! Although it does not rate highly on many of the dimensions of complexity described in chapter 1, the fast-food environment is sufficiently open to disturbances that instruction-based task analyses lead to the design of ineffective information systems. Moreover, this finding is not an isolated one. There are other documented cases showing that computer information systems based on prescriptive rationalizations of how tasks should be conducted either go unused or are ineffective in open systems with unpredictable disturbances (e.g., Guerlain, 1995; Hirschhorn, 1984; Roth et al., 1987; Sachs, 1995).

The bottom line is that instruction-based forms of task analysis are not very useful for appreciably open systems. If this limitation is ignored, then the resulting design will range from being not as effective as it could be to unusable, as in the case of the fast-food example. And as we mentioned, most existing task analysis techniques are of the instruction-based type and thus not ideal for our purposes. What about constraint-based forms of task analysis? Can they meet the challenges of open systems?

Constraint-Based Approaches: A Partial Savior

Fortunately, constraint-based approaches to task analysis can accommodate some of the context-conditioned variability required to achieve goals in open systems. As we discussed earlier, that is one of the primary advantages of constraint-based analyses—they are flexible enough to accommodate variability in worker action. Because they specify only the goal to be achieved and the constraints on achieving that goal, such analyses are indifferent to variations in precisely how the goal is to be achieved. In fact, both of the negative feedback models discussed earlier (see Figs. 3.5 and 3.6) are actually constraint-based structures, not instruction-based. There is no preplanned set of actions built into the control loops. On the contrary, the actions that are required to achieve the goal *emerge* on the spot in real time as a function of the

initial conditions and disturbances of the moment. In other words, the negative feedback models are capable of a very elementary form of context-conditioned variability (or situated action; Suchman, 1987).

The only factors that remain constant in the control loop are the goal to be achieved, and the constraints on goal achievement. Constraint-based approaches provide only this level of guidance. For closed systems, this vagueness could be a disadvantage. It might be preferable to minimize errors caused by human information-processing limitations by providing more detailed guidance (e.g., a flow sequence or a timeline). However, the preceding analysis shows that this vagueness is actually an advantage for open systems. The only way to adapt to disturbances is to leave some decisions about how the task should be performed to the worker, rather than to specify all of the details up front based on idealized (and unrealistic) assumptions about what should be done.

It is extremely important to note, however, that *discretion is not the same as complete freedom.* This is a point that is frequently overlooked, and therefore, a common cause of misunderstanding of the ideas presented in this book. We are not advocating that workers be allowed to do whatever they want. This would be irresponsible because it would include unintentional errors ("it seemed like a good thing to do at the time"), not to mention deliberate acts of sabotage. The discretion and flexibility that we are advocating is bounded by constraints. Errors and acts of sabotage fall into the areas of the state space that workers are required to avoid. Thus, the position we are advocating is to give workers discretion *within the boundaries of safe and effective operation.* Does this mean that constraint-based task analyses are the work analysis technique of choice for complex sociotechnical systems? Unfortunately not.

Taking Stock and Looking Forward: An Unresolved Problem

Recall that we began this chapter with the goal of determining if normative work analyses could meet the challenges imposed by complex sociotechnical systems. Task analysis was chosen as a prototypical example of such normative methods, and several different techniques were described. We realized that these different techniques could be organized into two categories, constraint based and instruction based. Instruction-based techniques are, by far, the most numerous, so we might expect that they would be more useful. We examined the viability of each of these categories of task analysis for complex sociotechnical systems by adopting a control-theoretic perspective. Our conclusion was that instruction-based analyses are not very useful for systems that are substantially open to unpredictable disturbances. Complex sociotechnical systems clearly fall into this category, as do many other simpler work environments.

These conclusions were reinforced by the case study of an information system installed in fast-food restaurants. The computer aid observed there seemed to be designed according to an instruction-based analysis of the task. Because the work environment was subject to unpredictable disturbances, workers did not use the aid for the purposes for which it was intended. Empirical data from other studies cited earlier reinforce the same point—instruction-based task analyses lead to brittle computer systems that cannot accommodate the context-conditioned variability that

is required for efficient and successful performance in open systems. And because most existing task analysis techniques are of the instruction-based variety, they are not well suited to our purposes.

Constraint-based forms of task analysis provide an attractive alternative. They give workers more discretion, thereby fostering better worker health (Karasek & Theorell, 1990). They can lead to new design functionality, making it more likely that productivity will be improved (Landauer, 1995). They also accommodate more variability in worker action, providing the flexibility required to adapt to unpredictable disturbances and increased opportunities for learning. Thus, constraint-based forms of task analysis are well suited to the demands of complex sociotechnical systems, and go partway toward addressing the challenges imposed by such systems.

However, they are not capable of meeting all of those challenges. This important qualification can be recognized if we uncover an implicit assumption that is buried in the two control-theoretic models presented earlier and the case study from the fast-food industry. In all three of these cases, it has been assumed that the goal to be achieved is well defined and can be established ahead of time, even though the sequence and timing of actions that are required to achieve the goal cannot be predicted. But there are some situations in complex sociotechnical systems where even the particular goal to be achieved may not be identifiable beforehand. The prototypical example is an unanticipated emergency in a nuclear power plant (e.g., the TMI accident discussed in chap. 2). If the event that triggers the emergency cannot be known beforehand, how can we know what the goal should be? Although it is not widely recognized, this problem can exist in many comparatively more mundane situations. A good example was provided by Shepherd (1992) in an application of task analysis to maintenance training for mechanical fitters in a chemical company: "A major problem that had to be dealt with at the outset was that fitters, like most craftsmen, are supposed to do anything they are called upon to do: there is apparently no one task to analyze" (p. 328).

One way to deal with this dilemma is to specify the task goal at a high (i.e., vague) level. For example, Shepherd (1992) identified one task goal as requiring workers to "Note any problems" (p. 335). This tactic tries to overcome ignorance about the precise goal to be pursued by specifying a very broad goal that subsumes all of the more specific goals that might be associated with unanticipated events. The problem with this approach is that it merely provides a place holder for what workers are supposed to do. It does very little to identify the information or knowledge that workers require to cope with the novelty imposed by unanticipated events.

The crux of the problem seems to be that all task analysis methods are *event-dependent* (Vicente & Tanabe, 1993). They require a specification (or assumption) of at least a class of initiating events before the analysis can even get off the ground. Otherwise, the precise goal to be pursued cannot be identified. Therefore, even constraint-based forms of task analysis do not provide a very satisfactory basis for dealing with unanticipated events.

This is an unacceptable situation for complex sociotechnical systems because, as we mentioned in chapter 1, unanticipated events pose the greatest risk to safety in such systems. We need some other form of work analysis if we are to support workers in these challenging situations. More specifically, we need *event-independent* work analysis techniques (Vicente & Tanabe, 1993) whose relevance and utility are not tied to a specific, finite class of anticipated events. In the next section, we develop a better idea of how this need can be met.

TABLE 3.2 Relative Advantages and Disadvantages of Directions and Maps

	Directions	Maps
Mental economy	efficient	effortful
Ability to adapt to unforeseen contingencies	brittle	flexible
Scope of applicability	narrow	broad
Ability to recover from errors	limited	great

DEALING WITH UNANTICIPATED EVENTS: THE VIEW FROM SPATIAL NAVIGATION

In this section, we derive some insights about how to conduct an event-independent work analysis by drawing on a problem with which everyone is familiar, namely spatial navigation. This analogy reveals the power of *work domain analysis* (see Footnote 1), a complementary form of work analysis that overcomes the weaknesses of task analysis.

Directions Versus Maps

Thorndyke and Goldin (1983) investigated how people learned to find their way in their everyday environments (e.g., while walking or driving in a city). Among the types of spatial knowledge they identified were procedural knowledge and survey knowledge.[9] *Procedural knowledge* represents the sequence of actions that people are required to take to follow a particular route. An example would be a set of directions to get from your home to a friend's home by rote (e.g., Go up highway 427 northbound, take the Rathburn Road exit west, keep going straight for about 2 km, turn left at Mill Road, etc.). *Survey knowledge* represents the spatial relationships between locations and routes in an environment. An example would be someone who has bird's-eye-view knowledge of the relative location of the streets in their neighborhood. Thus, procedural knowledge tells you what to do, whereas survey knowledge just tells you about the layout of the land. These two types of knowledge lead to the design of two complementary types of navigation aids. Procedural knowledge can be embedded in *directions*, and survey knowledge can be embedded in a *map*.

Strengths and Weaknesses. These two types of spatial aids have complementary advantages and disadvantages, as shown in Table 3.2. Instructions are more efficient because they tell you what you have to do. This mental economy can be achieved only because someone (e.g., the person who gave you the directions) was able to derive an appropriate set of actions to get from a particular starting point to a particular destination. They had to conduct this "analysis" beforehand and only then could they have built in the insights gained from that analysis into a set of

[9]The third form of spatial representation identified by Thorndyke and Goldin (1983) was landmark knowledge.

directions. Because they have already done most of the thinking for us, all we have to do is follow the actions in the directions. In contrast, with a map we have to do the thinking ourselves to derive a particular route from where we are to where we want to be. As a result, navigating with a map is less efficient than navigating with directions in situations for which directions are available.

However, this decrease in efficiency is compensated for by an increase in both flexibility and generality. Maps are more flexible because they allow us to derive different routes to get to the same location. This is particularly useful when an unforeseen event occurs. For example, if we are driving home and find that there is an accident on our regular route, causing a tremendous traffic jam, we can use our map to adapt online by deriving a new route to our destination that avoids the traffic jam. In contrast, if we only had directions, we would not be able to adapt to this unforeseen contingency. All we would have is a procedure for how to get from one point to another using a particular route. As soon as we deviate from that route, our directions would be of little use because they are tailored to that one sequence of actions. Thus, although they may be less economic, maps are more flexible (or conversely, less brittle) than directions.

Maps also have another related advantage, namely their generality. Because they are analogical representations (Woods, 1998), maps contain all of the possible destinations and starting points for a given geographical area (assuming they are comprehensive and accurate).[10] This property provides a tremendous amount of generality. It allows us to derive a route to get from any one point to any other point. In other words, maps are event-independent in the sense that their applicability is not tied to the particular starting or destination points that comprise a particular navigation event. This is an obvious advantage, especially if we frequently have to deal with novel travel plans. For example, if we are salespeople and we have to get to the same destination (e.g., our home) from a new starting point at the end of each day (e.g., a different customer's place of business), then maps provide us with the type of representation that we need to accommodate this variability in task demands. Similarly, if we leave from the same starting point at the start of each day (our home again), but have to drive to different destinations every day, then maps offer the same powerful generality. This provides a stark contrast to the limited possibilities made available by directions. Because they are tailored to one starting point, one route, and one destination, directions have a much more limited scope of applicability.

This observation also has important implications for the ability to recover from errors. If all we have is a set of directions, as soon as we make a mistake our navigational aid is useless, unless we can get back onto the route embedded in our directions or unless the particular error has been anticipated and built into the directions (e.g., "if you get to the bridge, you have gone too far"). In contrast, if we have an accurate and comprehensive map, recovery from errors is effortful, but possible. It is effortful because it requires some replanning to compensate for the error and get back on track. It is possible, however, because the utility of the map

[10]You may be wondering what happens if a map is out of date or contains an error. One of the benefits of analogical representations, of which maps are an example, is that they contain a great deal of redundancy (e.g., the position of any one object can be cross-referenced from many different locations). This redundancy can actually allow people to discover errors in the representation through a process of converging operations.

is independent of our current location (as long as the map covers the area of interest). Thus, we can determine how to get from where we mistakenly wound up to where we really want to go to.

Relevance to Work Analysis

The distinction between these two forms of navigational aids—directions and maps—has an analog in two forms of work analysis. Like directions, *task* representations tell workers what goals they should be trying to achieve, or also how they should be achieving them (see previous discussion). Like maps, *work domain* representations merely describe the structure of the controlled system (see chap. 7 for more details). More intuitively, a task is *what* workers do, whereas a work domain is what workers do it *on* (i.e., the object of action). For those familiar with computer programming, this distinction is similar to that between a control structure and a data structure, because the control structure operates on the data structure.

Table 3.3 shows that the comparative advantages and disadvantages of directions and maps are analogous to those between task and work domain analyses. Task analyses are efficient because they identify what needs to be done (in the case of constraint-based analyses), and perhaps even how it should be done (in the case of instruction-based analyses). But, as a result of this specificity, task analyses do not provide the support required to adapt to unanticipated events (see earlier discussion). Task analyses are also narrow in their generality because they are applicable only to the tasks that have been identified up front (as in the case of constraint-based analyses), or even more narrowly, to the particular ways of doing the task that have been identified up front (as in the case of instruction-based analyses). Finally, and as a result, task analyses are also limited in their ability to support recovery from errors.

As shown in Table 3.3, work domain analyses have a complementary set of strengths and weaknesses. Their primary disadvantage is that they do not tell workers what to do. They merely describe the capabilities of the system that workers will be acting on. As a result, work domain analyses put greater demands on workers for situations in which appropriate task descriptions are available. The good news is that work analyses are flexible because they provide workers with the information they need to generate an appropriate response, online in real time, to events that have not been anticipated by designers. Moreover, work domain analyses also have a broader scope of applicability. Because they merely show what the work domain is capable of doing—independent of any particular event—they provide workers

TABLE 3.3 Relative Advantages and Disadvantages of Task Analysis and Work Domain Analysis

	Task Analysis	Work Domain Analysis
Mental economy	efficient	effortful
Ability to adapt to unforeseen contingencies	brittle	flexible
Scope of applicability	narrow	broad
Ability to recover from errors	limited	great

with the discretion to meet the demands of the job in a variety of ways that suit their preferences or the particular needs of the moment. Finally, for the reasons already discussed, work domain analyses also provide workers with the support they need to recover from errors.

Conclusions

Two points are worth highlighting from this comparison. First, work domain analyses are absolutely essential for complex sociotechnical systems. They provide a way of supporting worker adaptation to novelty, thereby addressing the criterion of safety, and they provide discretion—not complete freedom—to workers, thereby addressing the criterion of health. Second, because they have complementary strengths and weaknesses, it would be useful to include both work domain analysis and constraint-based task analysis techniques in a single, integrated framework for work analysis. The framework described in the remainder of this book achieves this goal by including a work domain analysis phase (see chap. 7) and a constraint-based task analysis phase (see chap. 8) in one integrated, overarching framework.

SUMMARY

What type of work analysis is appropriate for complex sociotechnical systems? In this chapter, we evaluated the suitability of normative approaches to work analysis in the form of task analysis. We learned that the vast majority of existing task analysis techniques are instruction based. Yet, such techniques are not well suited for complex sociotechnical systems because they underestimate context-conditioned variability, and if used, can lead to unusable or ineffective computer information systems. Thus, most existing task analysis techniques are not very useful for our purposes. Constraint-based task analysis techniques are better suited for complex sociotechnical systems. Moreover, they are more likely to lead to improvements in flexibility, productivity, worker health, and on-the-job learning. However, they are not capable of dealing with the demands imposed by unanticipated events. Fortunately, work domain analyses provide a basis for dealing with such events. Thus, a work analysis framework for complex sociotechnical systems should include both work domain analysis and constraint-based task analysis techniques. The bottom line is that constraint-based task analysis techniques are necessary, but they are far from sufficient. In the next chapter, we see if additional insights can be garnered from descriptive approaches to work analysis.

Descriptive Approaches to Work Analysis: "What Workers Really Do"

4

> The principle of flexibility creates a conception of work in which the worker's capacity to learn, to adapt, and to regulate the evolving [automation] becomes central to the machine system's developmental potential.
>
> —Hirschhorn (1984, p. 58)

> Innovation = imagination of what could be based on knowledge of what is.
>
> —Lucy A. Suchman (personal communication, April 1997)

PURPOSE

In chapter 3, we reviewed normative approaches to work analysis to see how well they stand up to the unique challenges imposed by complex sociotechnical systems. In this chapter, we do the same for descriptive approaches to work analysis. The important contributions of descriptive approaches are illustrated through four case studies. The convergent findings from these cases help us identify additional dimensions that must be considered in work analysis. We argue that these descriptive techniques are very important and useful in understanding what workers really do and what they would like to do. Nevertheless, there are limitations in extracting design implications from descriptive approaches to work analysis. Our conclusion is that the descriptive analysis of current practice should be viewed as one of several possible means to investigate intrinsic work constraints, rather than an end in itself. Computer-based information systems should not be designed based solely on studies of current practice, nor should they be designed to support just the practices in which workers are currently engaged. These insights directly motivate a need for the formative approach to work analysis introduced in the following chapter.

DESCRIPTIVE APPROACHES: CURRENT PRACTICE

Scope

Descriptive approaches to work analysis are qualitatively different from normative approaches. Rather than postulating "rational" benchmarks for how workers should behave, descriptive approaches seek to understand how workers actually behave in practice. This goal is accomplished by conducting field studies that document the (usually quite dynamic) practical challenges that workers actually face on the job, and the (usually quite ingenious) practices that workers have developed to cope with those challenges. Thus, descriptive approaches to work analysis are called

descriptive because they seek to document and understand current practice in order to suggest ideas for new designs.

In North America, descriptive work analyses, particularly field observations in naturalistic settings, are frequently seen as a relatively recent innovation. However, there is a rich tradition of field study research in Europe going back at least 30 years (e.g., see some of the selections in Edwards & Lees, 1974, as well as Pejtersen, 1973; Rasmussen & Jensen, 1973). The Francophone ergonomics community, in particular, has placed great emphasis on phenomenological descriptions of current practice in naturalistic work settings (for overviews in English, see De Keyser, 1991; De Keyser, Decortis, & Van Daele, 1988). This research tradition makes a strong distinction between task and activity (e.g., Leplat, 1989, 1990) that is equivalent to the distinction between normative and descriptive approaches to work analysis. The term *task* refers to the official actions that are prescribed to workers, and is thus representative of the normative approach. The term *activity*, on the other hand, refers to the informal actions that workers actually perform in practice, and is thus representative of the descriptive approach. This distinction is at the very heart of Francophone ergonomics, and shows that descriptive approaches to work analysis have a relatively long history, at least in Europe.

As with normative approaches, there are many different perspectives that can be categorized as being descriptive. A comprehensive review of this body of research would require a book of its own. Therefore, we limit ourselves to describing four prominent case studies of descriptive work analyses conducted by researchers from different disciplines. Despite the diversity in the background of the analysts and in the application domains, the findings from these studies exhibit a surprising degree of convergence.

What Is Current Practice?

A Case Study From Situated Action. Suchman (1987), an anthropologist, conducted a representative study (cf. Brunswik, 1956) examining how pairs of office workers cooperated and interacted with a prototype expert help system for a photocopying machine. She used conversation analysis—a perspective for understanding face-to-face communication between people—as a basis for understanding communication between workers and computers. A key insight from conversation analysis is that

> Communication succeeds in the face of . . . disturbances not because we predict reliably what will happen and thereby avoid problems, or even because we encounter problems that we have anticipated in advance, but because we work, moment by moment, to identify inevitable troubles that arise. (p. 83)

These were the same insights that we derived in generic form from the negative feedback models in chapter 3. In an open system with external disturbances, workers must exhibit context-conditioned variability if they are to accomplish task goals.

The theoretical foil (i.e., antagonist) that Suchman (1987) was arguing against in her study was the artificial intelligence approach to planning that had been frequently adopted by cognitive science researchers. This view assumes that detailed plans can be derived, a priori, to specify what actions must be taken to accomplish particular

tasks. Essentially, this is the same perspective that is embodied in the instruction-based approaches to task analysis discussed in chapter 3.

Suchman (1987) compared these two views of human activity by studying how workers interacted with a computer system that was designed according to this very detailed instruction-based view of action. She videotaped office workers' first encounters with a computerized expert help system that was intended to help them operate a large and complex photocopier. In each session, two workers, neither of whom had used the expert system before, collaborated in pairs to perform representative photocopying tasks. The interactions between the workers and the expert system, and between the workers themselves, were logged, transcribed, and analyzed.

As we would suspect from chapter 3, Suchman (1987) observed that the success of an expert help system designed according to an instruction-based approach is "constrained by limitations on the designers' ability to predict any user's actions" (p. 120). The results of her study showed that this condition cannot be satisfied, even in the comparatively simple situation of two office workers interacting for the first time with a photocopier. Workers found alternate, equally plausible interpretations of the messages presented by both the help system and the photocopier, interpretations that unfortunately were not aligned with those that the designers intended. Furthermore, workers encountered contingencies that were not included in the detailed plans built into the expert help system. Consequently, the advice offered by the system was no longer appropriate for the unanticipated situation in which the workers found themselves, much like the computer "aid" for fast-food restaurants described in chapter 3. Needless to say, workers became confused and performance was far from efficient.

Suchman's (1987) conclusions are very similar to the ones we expressed in chapter 3. In open systems, context-conditioned variability is required to get the job done. In Suchman's own words, "purposeful actions are inevitably *situated actions*" (p. viii, emphasis in original) that are responding to "local interactions contingent on the actor's particular circumstances" (p. 28). Thus, the actual behavior of workers in situ is quite different from the rationalized ideal set forth by normative techniques for work analysis.

A Case Study From Naturalistic Decision Making. Klein (1989), an experimental psychologist, described several retrospective naturalistic studies of current practice that he conducted with his colleagues. Their first project investigated how fire-fighting commanders reported making decisions under stress. Klein and colleagues adopted a phenomenological approach, interviewing highly experienced commanders to learn how they made decisions in dynamic situations involving time pressure, risk, and personal accountability. In one study, the interviewees had an average of 23 years of job experience. During the interviews, they were asked to describe critical, nonroutine incidents that presented them with particularly challenging decision problems.

The theoretical foil for Klein's (1989) study was classic decision theory, which claims that workers should follow a thorough and "rational" approach if they are to make good decisions. For example, multiattribute utility theory models (e.g., Raiffa, 1968) recommend that workers: generate the set of options that are available, identify an appropriate set of evaluation dimensions, attach weights to each of these dimensions, rate each option on each dimension, and then select the option with the

highest utility score. How well do these prescriptive approaches account for the decision-making behavior reported by the fire-fighting commanders interviewed by Klein?

Consistent with Simon's (1956) much earlier observations, Klein (1989) found that the normative canons prescribed by the classic approach to decision making did not do a very good job of accounting for the retrospective interview data he collected. This led him to propose *recognition-primed decisionmaking* (RPD), a descriptive model of how experts frequently make decisions under time pressure and risk (see Klein, 1989, for a complete description). The RPD model provides a strong contrast to normative decision models. Rather than generating many alternatives in parallel, assigning weights, and comparing options, the fire-fighting commanders reported that they were instead acting and reacting to the evolving situation on the basis of their experience. Why did experts "go with the flow" rather than follow the prescriptions of the normative approach?

Just as there were good reasons for the nuclear power plant operators in our story to deviate from their procedures (see Preface), there were also several good reasons why the fire fighters did not follow the canons of normative decision models. First, there was little time available to choose a course of action, so it would be very difficult to engage in the laborious mental deliberations recommended by prescriptive, analytical approaches. As a result, the fire-fighting commanders did not try to find an optimal action, but instead focused on identifying a course of action that was feasible, timely, and cost-effective. In Simon's (1956) terms, they satisficed rather than optimized. Second, and more important, the commanders were not novices at the job but instead had a great deal of experience that they could exploit. Consequently, they were able to rely on their well-tuned perceptual capabilities to recognize informative cues in the environment. Given these cues, the commanders could selectively and efficiently recognize a particular situation as belonging to a category with which they were familiar. And because they were able to recognize the situation, they also knew what actions were appropriate, based on what had worked in the past under similar circumstances. By relying on their experience, the first option that fire-fighting commanders identified frequently turned out to be a workable action. Rather than have to generate all possible action alternatives up front—including many that, on the basis of experience, would be clearly inappropriate for the current situation—commanders could quickly identify a single viable option that frequently allowed them to get on with doing their job. As one fire fighter put it, "We don't make decisions. We put out fires."

In summary, Klein found that expert fire-fighting commanders were able to exploit their experience base to recognize situations in an efficient and timely manner, and to directly associate that situation assessment with a relevant action. Subsequent studies in other domains have often led to similar results (see Klein, 1989, 1997, for reviews). These findings thereby lend some generalizability to RPD, "a descriptive model" of expert decision making (Klein, 1989, p. 85). They show that the way in which workers actually made decisions was quite different from the normative approach prescribed by classic decision theory.

A Case Study From Activity Theory. Bødker (1991), a computer scientist, was involved in a number of projects involving the participatory design of computer-based systems. The largest of these was the UTOPIA project, "a Scandinavian research

project on trade union based development of and training in computer technology and work organization, especially text and image processing in the graphic industries" (Bødker, 1991, p. 7). As part of this project, descriptive work analyses were conducted to understand how workers currently achieved task goals. Based on these analyses, requirement specifications for future devices were developed and then prototypes were created and tested. Workers actively participated in each of these phases.

According to Bannon and Bødker (1991), the theoretical foil for Bødker's (1991) study was the Anglo-American approach to HCI. Although things have changed somewhat in the last 5 years or so, this research tradition had focused primarily on the artifacts being designed, with comparatively little regard for the context in which the artifacts were going to be used. For example, much more attention was paid to the syntactic and lexical features of the interface (e.g., Should windows be tiled or overlapping? How broad and deep should a menu be?) than to the semantics of the application domain. This emphasis could also be seen in the importance placed on usability—designing computer-based systems that are easy for workers to use—and the comparative lack of attention to usefulness—designing the functionality that is required to do a particular job effectively (cf. Landauer, 1995). Referring back to chapter 2, we can say that Anglo-American HCI was much more influenced by the cognitivist perspective than by the ecological perspective.

While conducting descriptive analyses of work during the UTOPIA project, Bødker (1991) concluded that the Anglo-American approach to HCI did not provide an accurate or comprehensive account of her findings. The traditional paradigm seemed too narrow and superficial to account for the richness and complexity of the practices exhibited by workers. This led Bødker to search for an alternative theoretical framework that was capable of explaining her descriptive findings. Note that Bødker was not content to merely document what she observed. She also wanted to identify a systematic framework consisting of well-defined concepts that could be meaningfully applied in naturalistic settings (cf. the motivation for our Glossary).

This quest led Bødker (1991) to *activity theory,* a psychological theory that had been developed by basic researchers in the Soviet Union. Bødker found that activity theory could be adapted to the practical needs of HCI, and more important, that its concepts provided a good descriptive understanding of the data collected during UTOPIA and other projects. Although a comprehensive account of activity theory is well beyond the scope of this chapter, we can briefly describe some of its characteristics to illustrate the general nature of the theory (see Bedny & Meister, 1997; Nardi, 1996b, for more theoretical details and sample applications). The core focus of activity theory is the study of goal-directed activity. As simple as this idea may appear, in the context of HCI it results in a profound shift in emphasis. Instead of focusing primarily on the device, activity theory focuses primarily on the goals that workers are trying to satisfy with the device. In this view, the device is merely a means, not an end in itself. Thus, rather than viewing HCI as working with computers, activity theory views HCI as working through computers.

This change in emphasis has several important implications. First, it requires a deep understanding of the semantics of the domain (see chap. 1). Rather than just focusing on superficial surface features of the device, analysts have to spend much more time understanding the application domain. Otherwise, they will never be able to understand workers' activities. Second, goal-directed activity is usually conducted with tools in a social context that has a cultural history, at least in naturalistic settings. Consequently, the scope of HCI must be considerably broadened to accommodate these

additional factors. To understand activity, we must understand how tools influence current practices, a relationship we address in more detail later. We must also understand how people interact with, and learn from, one another. Thus, social and organizational issues come to the fore. In addition, we must understand how current practice has been influenced by cultural history. Decisions made in the past, and influences of the past, shape current practice. Thus, the development of these changes over time can itself become the object of inquiry. Third, because of its emphasis on studying activity in situ, activity theory puts a premium on understanding human expertise and skill. This characteristic is valuable in HCI applications where we want to construct computer-based systems that enhance and foster worker competence. Finally, as a result of all of these factors, activity theory views action as situated. Workers' actions are adapted to the current context. Thus, once again, we find the importance of context-conditioned variability in understanding workers' actions in open systems.

These general characteristics are represented in activity theory by a number of systematic concepts, such as: activity, action, operation, object, tool, subject, and community. According to Bødker (1991) and others (see Nardi, 1996b) these concepts, with their focus on practice, provide a much broader frame of reference for HCI than the traditional Anglo-American approach.

A Case Study From Distributed Cognition. Hutchins (1995a), an anthropologist, conducted a field study of current practice in the domain of ship navigation. He collected his data as an observer, primarily on the navigation bridge of a number of U.S. Navy ships at sea. During this time, he made detailed records of workers' actions and communications using written notes, audiotapes, and eventually, a video camera. The recordings were subsequently transcribed and analyzed based, in part, on Hutchins' extensive experience with, and knowledge of, the navigation task. The goal of these investigations was to better understand the nature of human cognition in naturalistic settings.

The theoretical foil for Hutchins' (1995a) work was the physical symbol system hypothesis (PSSH) put forth by A. Newell and Simon (1972). The PSSH is based primarily on findings from experiments with inexperienced people performing laboratory puzzles and games (e.g., cryptarithmetic, chess, and Tower of Hanoi). Note that these are closed systems (see chap. 3). Furthermore, in these experiments, people are typically faced with novel and thus, mentally challenging, situations. Nevertheless, people are usually forced to work alone and are usually not allowed to use any aids (e.g., pencil and paper) to solve the task at hand. The view that has developed from these studies is that human cognition involves mental information processing driven by well-defined rules and representations stored in human memory. In essence, human cognition is likened to a digital computer.

Hutchins' (1995a) naturalistic observations of current practice led him to a very different view of human cognition. First, he found that knowledge and information processing are not confined to the brain, but are instead distributed spatially across individuals and artifacts and temporally as a function of the history of a particular culture. Workers were rarely observed in the isolated, pensive activities that we would expect based on the PSSH. Instead, they frequently relied on external artifacts to reduce the burden on their limited cognitive resources, thereby allowing them to achieve task goals in a more economic fashion. Second, workers frequently accomplished task goals, not in isolation through mental information processing, but as a

functional team through mutual coordination of their actions. As a result, team communication played a very important role. Furthermore, the redundancy achieved by having multiple people involved in performing a task also had the added benefit of creating a robust mechanism for error detection and error recovery. When one worker makes an error, it is frequently noticed by another because team members have access to each other's actions and communications. Third, cognition "in the wild" is an emergent activity that is not completely specified ahead of time, whether it be by written procedures or by mental rules and representations. Instead, the pattern of coordinated action exhibited by teams are generated anew each time "without there being a representation of that overall pattern anywhere in the system" (Hutchins, 1995a, p. 200). Using the terms we introduced in chapter 3, cognition is situated and exhibits context-conditioned variability. Finally, Hutchins also found that the historical influence of culture was very important. Many of the useful artifacts that workers rely on (e.g., navigation charts) and many of the practices in which workers are engaged (e.g., simplifying rules for calculation) were adaptive products of hundreds of years of navigational experience. In short, cognition in naturalistic settings is not a strictly mental activity mechanically performed by an individual in isolation using mental rules and representations. Instead, it is an emergent, distributed activity that is performed by people with tools, within the context of a team or organization, in a cultural context that has evolved over years of experience.

It is important to note that not all of the work practices that Hutchins (1995a) observed were documented as official practice. Although there were procedures that were said to prescribe how the navigation task is supposed to be performed, "the normative procedures are an ideal that is seldom achieved, or seldom achieved as described" (Hutchins, 1995a, p. 28). Thus, like the other descriptive analyses reviewed earlier, Hutchins found that normative descriptions of work (i.e., the task, in Francophone ergonomics) do not correspond to the practices in which workers are actually engaged (i.e., the activity, in Francophone ergonomics). Thus, yet again, we see the value added by descriptive approaches to work analysis.

The Importance of Studying Current Practice

Table 4.1 summarizes these four case studies of descriptive work analysis. There are several differences between the four cases. First, they were conducted by researchers from largely disparate backgrounds. Second, they focused on very distinct application domains. Third, their respective theoretical foils also differed, at least at a superficial level. Despite these substantial dissimilarities, the cases are united by two overarching themes.

Common Themes. The first common thread is the enormous value of conducting descriptive studies of work in representative or naturalistic settings. As shown in Table 4.1, each of the four studies led to important insights that have influenced entire areas of research. Suchman's (1987) research is largely responsible for the increase in attention to anthropological methods and theories in the HCI community. Klein's (1989) research has influenced many psychologists and human factors researchers to study and better understand workers' expertise under naturalistic conditions. Bødker's (1991) work has greatly increased the profile of Soviet activity theory in the applied world of HCI. Finally, Hutchins' (1995a) research has had a

TABLE 4.1 Comparison of Four Case Studies of Descriptive Studies of Work Analysis

Case Study	Analyst's Background	Application Domain	Theoretical Foil	Major Contribution
Suchman (1987)	Anthropology	Photocopying	Artificial Intelligence	Situated Action
Klein (1989)	Experimental Psychology	Fire Fighting	Classical Decision Making	RPD Model
Bødker (1991)	Computer Science	Graphic Design	Anglo-American HCI	Activity Theory-Based HCI
Hutchins (1995a)	Anthropology	Ship Navigation	Physical Symbol System Hypothesis	Distributed Cognition

significant impact on basic and applied researchers in cognitive science by showing the value of distributed cognition for studying and shaping cognition. In each of these four cases, the researchers would not have been able to develop the same insights had they not ventured out into the field and meticulously analyzed and documented the varied and complex demands imposed on workers, and the ingenious adaptations that workers have developed to make those demands manageable. The picture of human work that emerges from these studies is very different from the one painted by the unrealistic assumptions made by the normative approaches to work analysis that we reviewed in chapter 3. Therefore, these four case studies collectively illustrate the "value added" by studying what workers really do.

The second unifying theme is the converging characterization of human work. First, although there are differences between the approaches, all four cases illustrate the importance of context-conditioned variability. Cognitive work in naturalistic settings takes place in an open system, and so people must frequently adapt to the contingencies of the moment. Second, work has a strong social component and rarely takes place in isolation. Workers are usually part of a team that must communicate and coordinate effectively to achieve their goals. Third, work is also seldom solely focused on internal mental processing because workers create tools (e.g., external representations) that are tailored to the work demands, thereby reducing the burden on scarce cognitive resources. Fourth, current work practices are shaped by historical cultural factors that have been ignored by many traditional approaches. Fifth, workers are frequently under time pressure and other constraints, so they must develop the expertise required to get the job done in the face of such pressures. Frequently, such expertise has a strong perceptual component, thereby reducing the load on cognitive resources. Based on all of these insights, we can make the following strong claim for open systems: *Workers do not, cannot, and should not consistently follow the detailed prescriptions of normative approaches.* Our story of "malicious procedural compliance" in the Preface is a vivid indicator of this point.

A set of articles in a recent special section of the *Communication of the ACM* devoted to representations of work emphasizes this conclusion (Suchman, 1995). These articles show that normative approaches to work analysis provide an inadequate basis for design because their assumptions ignore the richness and variability of actual work practices (see chap. 3). This point is an important one to make because normative approaches still dominate the design of computer information systems. In a more recent article, Simonsen and Kensing (1997) also advocated an examination of current work

practices as part of systems design. They explicitly stated that the approach they have adopted "develops descriptive understanding in contrast to prescriptive" (p. 82).

Implications for Work Analysis. The characterization of work emerging from these studies has important implications for a viable framework for work analysis in complex sociotechnical systems. If we include the insights from the previous chapter, we can conclude that such a framework must include at least five dimensions of work:

1. A *work domain analysis* is needed to identify the information requirements that will allow workers to deal with unfamiliar and unanticipated events (see chap. 3).
2. A *constraint-based task analysis* is needed to identify the information requirements that will help workers achieve anticipated task goals in a flexible, situated manner (see chap. 3).
3. An analysis of effective *strategies* is required to identify the mechanisms that can generate the practices that have emerged historically in the culture of a particular application domain. If our work analysis explicitly identifies such strategies, then designers can build information systems to support them.
4. Perhaps the most important lesson emerging from descriptive analyses of work is that we need to analyze the *social and organizational* factors that govern work. Although we speak of creating a computer-based information system, we are also inducing an organizational structure when we introduce information technology into the workplace. Thus, our work analysis should explicitly address the issue of how work can be allocated across individuals, how those individuals can be organized into groups or teams, and how individuals, groups, and teams can communicate with each other. By explicitly analyzing these social factors, we can try to coordinate technological change and organizational change.
5. Finally, it would also be useful to explicitly identify the various demands that the application domain imposes on individual *workers' competencies* (i.e., expertise). Identifying those demands would allow us to design computer-based information systems that facilitate skill acquisition and that support expert action.

WHY DESCRIPTIVE APPROACHES ARE NOT ENOUGH: THE TASK-ARTIFACT CYCLE

Although much can be gained by studying workers' current practices, it does not follow that computer-based information systems should be designed based only on descriptive studies, nor should they be designed to support only current practices. This is a subtle distinction that we believe has not received the attention it deserves. The point is a very important one, however, because it illustrates the limitations of descriptive approaches to work analysis and motivates a need for formative approaches.

Limitations of Basing Design Solely on Current Practice

The rationale behind our claim is illustrated in Fig. 4.1, which depicts two idealized, intersecting sets. The set on the left is defined by the constraints on achieving work goals, independent of any particular device. We have labeled this set *intrinsic work*

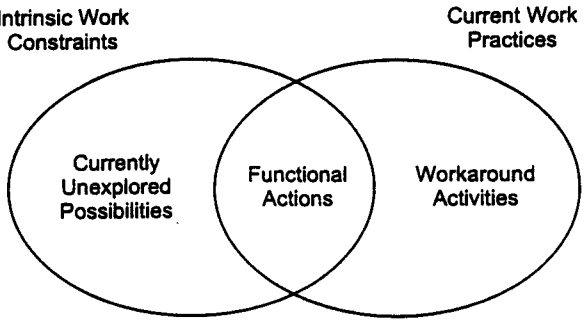

FIG. 4.1. Why computer-based information systems should not necessarily be designed to support current work practices. See text for details.

constraints to emphasize the fact that these constraints are an inherent part of work in a particular domain. This concept may seem like a Platonic ideal, so it is important to point out some examples: the position of mountains in air traffic control, foreign exchange rates in international financial markets, the laws of physics in a power plant, and the maneuvering capabilities of various enemy aircraft in military command and control. In each case, the constraints in question must be taken into account by any actor if task goals are to be reliably and consistently achieved, regardless of the device available. Thus, intrinsic work constraints are, in principle at least, device-independent because they delimit the actions that are required to get the job done, not the actions that are required to get the job done with a particular device. In subsequent chapters, we develop this concept in a more concrete and detailed manner.

The significance of intrinsic work constraints can be made clear by contrasting this set with the one on the right in Fig. 4.1, which is the set of actions that comprise *current work practices*. When we conduct a descriptive work analysis, we are observing a subset of the set on the right of Fig. 4.1. A crucial point, whose significance is discussed later, is that current practice is device-dependent because it represents the strategies and tasks that workers are using to get the job done with whatever tools they currently have at their disposal.

One important implication to emerge from this graphical representation is that these two sets are overlapping but not equivalent.[1] There are three qualitatively different regions that must be distinguished in Fig. 4.1. The first region is the intersection of the two sets, which represents the subset of actions that are intrinsic to getting the job done and that are a part of current work practice. These are *functional actions*, and their identification is a primary reason for conducting field studies of work under naturalistic conditions.

The second region in Fig. 4.1 is the subset on the far right, which is composed of actions that are not directly part of getting the job done, but that are nevertheless part of current practice. These *workaround activities* are a frequent source of inefficiencies in HCI. You may recall that Table 2.1 in chapter 2 provides a number of instances from the graphic design software investigated by Black (1990). For example, printing out one alternative draft at a time is not an inherent part of graphic design. It is an activity that workers have to engage in when the information system does not support the

[1]The relative size of, and the amount of overlap between, these two sets is unknown and surely varies from case to case. The depiction in the figure is intended to illustrate only qualitative distinctions, not quantitative differences.

demands of the job (in this case, comparing several initial design proposals in parallel before proceeding with the detailed design work). There are many other examples we could cite as belonging to this category (e.g., R. I. Cook & Woods, 1996). One omnipresent example in today's world of graphical user interfaces is the set of overhead activities associated with interface navigation and management. In some application programs, workers spend a great deal of their time and energy managing windows and other interface objects, such as dialogue boxes, messages, menus, and palettes (e.g., Harrison, Ishii, Vicente, & Buxton, 1995). In fact, when we videotaped workers interacting with such programs and then replayed the tape in fast-forward mode, what we saw was a furious level of activity with workers frequently moving interface objects, resizing them, bringing them up, and making them go away. All of these are workaround activities. They are not directed at accomplishing work objectives, at least not in a direct way. Instead, they are "overhead" activities that workers have to perform because of the impoverished interface they currently have.

Finally, the third region in Fig. 4.1 is the subset on the far left, which is composed of the actions that are an intrinsic part of work but that are not a part of current work practices. These are *currently unexplored possibilities* for getting the job done. They may be potentially very efficient and productive ways of accomplishing work. However, these actions are usually not a part of current practice because it is not feasible for workers to perform them with the current inadequate level of support with which they have been provided. For instance, currently unexplored actions may require too much time, computational effort, memory demands, or knowledge with the existing device. Returning to the domain of graphic design described by Black (1990), a hypothetical example could be the practice of comparing several alternative design proposals in parallel before going on to develop a detailed design. If this activity is cumbersome to accomplish with the existing information system (as it was with the software analyzed by Black), then workers may simply decide to omit this task. The result would be a reduction in product quality because of poor design practices. The general point is that workers sometimes do not exploit certain work domain possibilities, perform certain tasks, or adopt particular strategies because it would be too effortful or time consuming to do so with the tools they currently have available to do the job. However, these unexplored possibilities could very well become a productive part of workers' practices, if the requisite computer support were provided. In chapters 9 and 12, we provide detailed examples illustrating this very point.

Implications. This discussion is relevant to both design and work analysis. In terms of design, the implication is that computer-based information systems should ideally be designed to support intrinsic work constraints, not just current work practices.[2] As suggested by Fig. 4.1, there are two reasons to justify this claim. First, we do not want to base our design on the workaround activities that are vestiges of poor device design. And as Benyon (1992a) pointed out: "Current practice is *always* tied to existing technology. . . . [Thus,] embodying current practice in future systems is a fundamental error" (p. 114, emphasis in original). Thus, there is no

[2]This claim is valid only for the revolutionary design conditions we described in chapter 1 (cf. Alexander, 1964). There are many situations in industry where designers are forced to work strongly within the context of previous designs. In these cases, making assumptions about the existing device may be a good thing. However, it is important to realize that the resulting design will not be nearly as effective as it could be in these cases, because it is inheriting some of the deficiencies or limitations of the previous device.

reason why workaround activities should be supported by the new device (barring technological constraints). Second, we do want to support currently unexplored possibilities. There is no reason to exclude ways of doing the job that could be effective but that are too effortful or time consuming to adopt with the existing device (barring technological constraints). After all, such new functionality can lead to marked improvements in productivity (Landauer, 1995).

In terms of analysis, the conclusion of this discussion is that work analysis should focus on identifying intrinsic work constraints rather than just current practice (see Fig. 4.1). More specifically, the technique we choose for work analysis should try to explicitly "peel away" the subset of current practices that originate from poor device design, and it should explicitly try to uncover productive practices that can be used, not just those that are being used (Rasmussen et al., 1994). This does not mean that descriptive studies of current practice are not valuable. As the four cases reviewed earlier show, they are essential. Instead, the point is that future designs should go beyond current practice by removing unwanted inefficiencies and by adding new functional possibilities. The most direct way of achieving this goal is to adopt a work analysis technique that explicitly tries to identify intrinsic work constraints.

Are These Limitations Recognized?

The limitations we have identified have been explicitly recognized by a number of researchers from different backgrounds, including some who are advocates of descriptive approaches to work analysis. One example that we have already mentioned is that of Benyon (1992a), who stated that "embodying current practice in future systems is a fundamental error" (p. 114). More recently, Beyer and Holtzblatt (1998) echoed this opinion, stating that designing a new device to match existing work practices exactly "would be a sure path to failure" (p. 7). Holmqvist and Andersen (1991), who are strong advocates of descriptive work analyses, also pointed out that they "do not believe that a system designer should design the computer-based . . . system as a copy of the existing work" (pp. 91–92). T. E. Miller and Woods (1997), coming from a cognitive engineering perspective, voiced a similar opinion.

Thus, there are a number of researchers who believe that the design of future devices should not be based solely on current work practices. This is not to say that it is not valuable to study current work practices. On the contrary, as the overlap in the center of Fig. 4.1 shows, studying current practice can shed a great deal of light on intrinsic work constraints. However, the analysis of current practice should be viewed as one of several possible means to investigate work constraints, rather than an end in itself. This point can be better appreciated by examining how difficult it has been to move effectively from descriptive work analyses to implications for the design of computer-based information systems.

From Descriptive Analysis to Design Implications: The Track Record

Social Science. In a recent article, Button and Dourish (1996) observed that the attempt to incorporate the insights from anthropological analyses of current practice (specifically, those using an approach known as ethnomethodology) into the design of computer-based information systems has been "problematic" (p. 20). Descriptive

work analysis techniques have certainly been very effective in understanding how existing information systems fail to support human work. However, Button and Dourish stated that the attempt to design new information systems using these techniques "is fraught with methodological dangers" (p. 19). In some ways, this should not be surprising because ethnomethodology was not designed for such purposes. That "tradition is in analysing practice, rather than 'inventing the future' " (p. 21).

Similar, if less specific, concerns have been voiced by other social scientists. For example, Heath and Luff (1996) stated that:

> It is being increasingly argued that the requirements for complex systems need to be derived from a deeper understanding of real-world, technologically supported cooperative work, which in turn might lead to a distinctive, more social scientific, approach to user-centered design. *Whether or not such developments can be drawn from work in the social sciences is unclear.* (p. 98, emphasis added)

In summary, social scientists have contributed greatly by providing us with a better descriptive understanding of how work is conducted in situ (e.g., Engeström & Middleton, 1996). However, an additional body of knowledge and set of techniques are needed to design a better device.

Activity Theory. The same point can be illustrated in a more specific form by referring to the contributions to HCI based on activity theory documented in Nardi (1996b). There are few examples of novel designs based on an activity theory analysis (see Bødker, 1991, for an exception). There are at least two probable reasons for this. First, the application of activity theory to HCI is relatively recent, so there may not have been enough time for examples to emerge. Second, it is possible that the descriptive nature of activity theory makes it difficult to develop a novel design. This interpretation is supported by juxtaposing quotations from several chapters in Nardi's edited volume:

1. "Activity theory . . . proposes a very specific notion of context: the activity itself is the context" (Nardi, 1996c, p. 76).

2. "The activity is the way the subject sees the practice" (Christiansen, 1996, p. 176).

3. "The main strategy for research is simply to ask the persons to tell us about their action plans and goals" (Raeithel & Velichkovsky, 1996, p. 219).

4a. "Activities are always changing and developing" (Kuutti, 1996, p. 33).

4b. "Activity is, according to the theory, . . . a prism that moves and changes all the time" (Christiansen, 1996, p. 195).

These quotes illustrate that the focus of the work analysis is a moving target, making stable generalizations to new products difficult.

Like ethnomethodology (see previous discussion), activity theory was also not originally developed for design purposes. According to Nardi (1996a): "Activity theory focuses on *practice* . . . [and] is a powerful and clarifying *descriptive* tool rather than a strongly predictive theory. The object of activity theory is to *understand* the unity of consciousness and activity" (p. 7, emphasis added). Descriptive understanding is essential in determining why existing devices are ineffective or how workers' activities

have evolved over time. Our point is that this knowledge is useful, but not sufficient, for designing better information systems. Thus, despite its virtues, activity theory is subject to the limitations identified earlier. As a result, the claim that "activity theory . . . holds the best conceptual potential for studies of human-computer interaction" (Kaptelinin, 1996, p. 107) seems somewhat exaggerated. Different methods have different strength and weaknesses (cf. Weinberg, 1982), so no framework—including the one described in this book—can be considered to be "the best" overall.

Francophone Ergonomics. It is very important to point out that the concerns that we and others have voiced are not specific to work analysis techniques from the social sciences or from activity theory. They are instead deeply rooted in a fundamental limitation of descriptive approaches to work analysis. This conclusion is supported by the fact that other researchers, from independent traditions, have encountered identical problems in using this class of methods.

The Francophone ergonomics community, for example, has experienced this very same problem, despite having the benefit of over 30 years of experience with descriptive work analyses:

> One fact, however, seems certain. A wealth of data has been collected but the data are still in their raw state. It has not yet been possible to extract any formal models or generalizations from them. Hence, the difficulty of creating an ergonomics of design, which would enable us to proceed by methods other than empirical testing, trial and error or successive corrections. (De Keyser et al., 1988, p. 151)

Human Factors. The deep-rooted limitations of descriptive approaches to work analysis can also be observed in the human factors community. In chapter 3, we described how task analysis can be used in a normative fashion. It is also possible to use the same techniques to describe, not what workers should do, but what they are currently doing. To take but one example, the task analysis technique known as *link analysis* (Kirwan & Ainsworth, 1992) can be used to determine empirically the frequency with which workers interact with particular instruments or locations (e.g., the probability of visually scanning from one display to another, or the probability of walking from one area of a control room to another). Such task analyses suffer from the same problems as other descriptive techniques. For instance, though link analysis can help analysts identify bottlenecks in current practice, it is not nearly as effective in formatively specifying the attributes that a new device should possess if it is to eliminate or reduce those bottlenecks. Kirwan and Ainsworth provided a clear statement of the problem: "As the [link] analysis is very context-specific [i.e., device-dependent], it is not normally possible to add information, or make inferences, if the system changes . . . any changes will probably result in the analysis having to be totally restarted" (p. 123). The very same conclusion can be made for other descriptive task analysis techniques, such as activity sampling (Kirwan & Ainsworth, 1992). Thus, the limitations associated with descriptive approaches are not limited to work analysis techniques from any one discipline.

People Are Adaptive: The Task-Artifact Cycle

The problem is that the introduction of new technology is, to use Schmidt's (1991b) felicitous phrase, "like writing on water" (p. 76). Analyses of current practice lead to design ideas for supporting that practice. However, workers are adaptive, so the

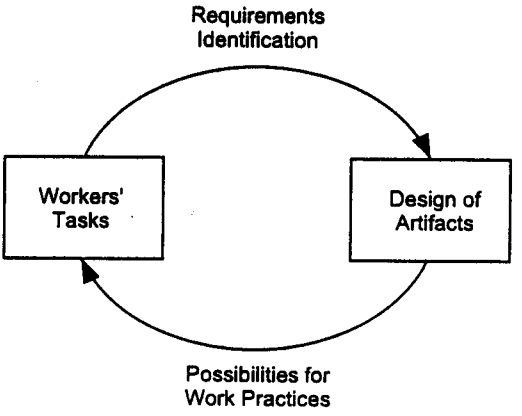

FIG. 4.2. The task-artifact cycle (adapted from Carroll et al., 1991).

introduction of a new design results in new practices that, more often than not, are not fully or well supported by the new device. The result is a new set of unforeseen problems (see Vicente & Williges, 1988, for an example). We have already encountered the tip of this iceberg in chapter 3 when we discussed the regress caused by the device-dependence of instruction-based approaches to task analysis.

This interdependence between current practice and the design of a device has been concisely captured in Carroll et al.'s (1991) task-artifact cycle, illustrated in Fig. 4.2. The circular nature of technological intervention is clearly shown. If we conduct a descriptive work analysis to understand workers' current tasks, we will identify requirements that could be used to design a new artifact. However, once this artifact is introduced into the workplace, new possibilities for work practices are created, thereby shaping workers' practices. In the words of Carroll et al.: "A task implicitly sets requirements for the development of artifacts to support it; an artifact suggests possibilities and introduces constraints that often radically redefine the task for which the artifact was originally developed" (p. 79). Thus, by basing new designs on work analyses of current practice, designers will always be one step behind their interventions.

Summary

In this section, we have tried to show why descriptive approaches to work analysis are valuable but limited in their ability to inform the design of computer-based information systems. Current practice reflects inadequacies and limitations in the existing device and, at the same time, hides potentially valuable but unexplored ways of working. These problems have been recognized by some researchers. Moreover, the difficulties in going from descriptive analysis to design implications have been encountered by a number of distinct research traditions that use descriptive methods for work analysis, including social sciences, activity theory, Francophone ergonomics, and traditional human factors. The root cause of these difficulties lies in the regress imposed by human adaptation, and described by the task-artifact cycle.

Our understanding of this problem suggests that work analysis techniques should seek to identify the set of intrinsic work constraints rather than just the set of current

work practices. This alternative approach should make it easier to move from analysis to design. Before we explore this view, however, we examine the techniques that other researchers have already developed to try to overcome the task-artifact cycle.

EXISTING TECHNIQUES FOR GETTING AROUND THE TASK-ARTIFACT CYCLE

The problems imposed by the task-artifact cycle have been recognized by at least some researchers for quite some time (e.g., Greenbaum & Kyng, 1991). Thus, it should not be surprising to find that at least two major techniques have been developed, and used, to try to get around the task-artifact cycle: rapid prototyping and scenario-based design. In this section, we review the valuable contributions of each of these techniques.

Rapid Prototyping and Iterative User Testing

One way to "go beyond the present interface" (Kyng, 1995, p. 49) is to create prototypes of new designs, evaluate them by having workers use them to perform representative tasks, and then use the evaluation findings to iterate on the design. The Scandinavian school of participatory design has made particularly effective use of this technique (e.g., Bødker, 1991; Greenbaum & Kyng, 1991). The rationale behind prototyping is straightforward, and can be explained with reference to Fig. 4.1. Instead of basing a design solely on descriptive work analyses of current practice, building and testing prototypes provides an opportunity to "dig into" the subset of currently unexplored possibilities. Each round of prototyping and testing can be used to evaluate hypotheses about what new functions might be useful to workers. Ideally, with each iteration, the subset of currently unexplored possibilities should become smaller and smaller, as valuable functionality is added (or changed) in each generation of prototypes. Similarly, building and testing prototypes also provides an opportunity to "dig out of" the subset of workaround activities. Each round of prototyping and testing can be used to evaluate hypotheses about deficiencies in the device that induce workarounds. Ideally, with each iteration, the subset of workaround activities should become smaller and smaller, as device deficiencies are removed in each generation of prototypes. Thus, the end goal is to iteratively maximize the overlap between the two sets in Fig. 4.1 through an active learning process involving both analysts and workers. At the end of the design life cycle, there should ideally be little distance between the intrinsic work constraints and the work practices induced by the new design. Bødker and Grønbœk (1996) provided a nice case study illustrating how prototypes can be used to try to overcome the inertia of current practice.

Scenario-Based Design

Whereas prototyping and testing provide an empirical way of trying get around the task-artifact cycle, scenario-based design (Carroll et al., 1991; Carroll & Rosson, 1992) provides an analytical technique for trying to achieve the same objective: "Scenarios

have the important property that they can be generated and developed even before the situation they describe has been created" (Carroll & Rosson, 1992, p. 190). Thus, rather than—or in addition to—building prototypes, it is possible to develop scenarios that envision what it might be like for workers to interact with a device that has yet to be designed. These scenarios can then be analyzed to identify the psychological consequences of particular device features in the situations defined by the scenarios generated by the analyst. These consequences can, in turn, be analyzed to determine how well the envisioned design supports workers in achieving task goals. Those insights can be iteratively used to build new scenarios and new envisioned artifacts. As with prototypes, the goal is to minimize the workaround activities induced by the new device and to incorporate new possibilities for functionality. In short, scenarios provide a means for analytically evaluating "simulated future work" (Kyng, 1995, p. 55). Carroll and Rosson provided several examples of how scenario-based design can be used to try to overcome the inertia of current practice.

Limitations: The Problems of Device-Dependence and Incompleteness

There can be no question that both prototypes and scenarios are valuable tools that can be effectively used in conjunction with the framework that we are advocating in this book. For us, the important question is: Are these techniques sufficient to overcome the task-artifact cycle when designing computer-based support for complex sociotechnical systems?

As valuable as they are, both prototyping and scenarios suffer from two limitations: strong device-dependence and incompleteness.[4] Regarding the former, prototyping and testing actually do not overcome the regress in the task-artifact cycle (see Fig. 4.2). Instead of escaping from the regress, these techniques actually involve iterating through the task-artifact cycle numerous times, with the expectation that the design will be improved with each iteration. Figure 4.3 illustrates this relationship graphically. Each prototype constitutes an artifact (e.g., A_1) that introduces a new set of possibilities for doing the task (e.g., T_1). Analysis of this task then identifies new requirements that, if adopted, lead to the design of a new artifact (e.g., A_2). Iteration through the cycle continues in this manner until the analyst is satisfied or (more typically) runs out of resources. Thus, iterating through the task-artifact cycle in this way is akin to a dog chasing its tail. Some advocates of these techniques acknowledge this point: "Generally, our approach should be thought of as an iterative one, where the introduction of the new computer application in the work setting may cause unforeseen changes in work, in turn leading to demands for new or changed computer applications" (Bødker, Greenbaum, & Kyng, 1991, p. 151). Note that Bødker et al. were referring to iteration in the workplace after a design has been introduced, not to the testing iterations that take place before a design is introduced into the workplace.

Because prototyping and testing do not avoid the task-artifact cycle, the requirements that are identified by analysts are limited by the choice of prototypes that are tested. However, there is no systematic way to go from results of testing to prototype

[4]Other techniques that have been proposed (Greenbaum & Kyng, 1991) for trying to get around the task-artifact cycle (e.g., mock-ups, Future Workshops, and metaphorical design), though useful, also suffer from the same limitations.

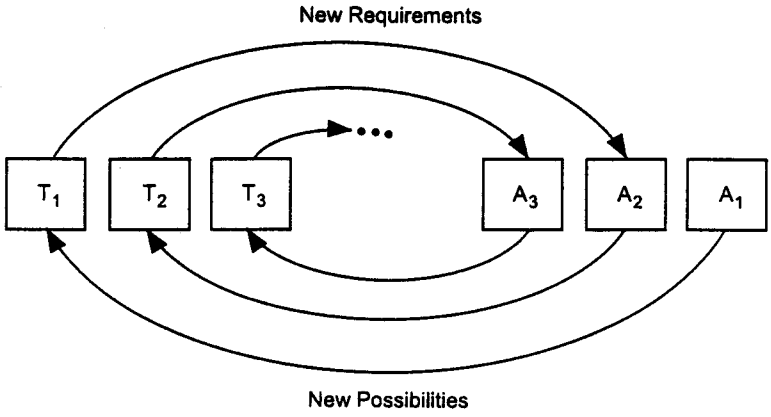

FIG. 4.3. Iterating through the task-artifact cycle. Compare with Figure 4.2 (T_i = $Task_i$; A_i = $Artifact_i$).

attributes. This latter step is largely limited by the ingenuity and creativity of the designer. Of course, creativity can never be eliminated from design (Ferguson, 1977), but the smaller we can make the gap between analysis and design, the less dependent we are on the skills of any one designer. Furthermore, any design change, even one occurring in a very late iteration, can lead to new, unforeseen problems. Thus, the aforementioned regress is never really avoided because the testing results are strongly device-dependent. Beyer and Holtzblatt (1998) encountered this problem in their experiences in the HCI industry: "Prototyping and usability testing could iterate an existing system, but couldn't suggest wholly new directions" (p. 20). Like us, they thereby felt a need to seek "a process that would lead to new kinds of systems rather than iterating existing systems" (p. 20).

Scenario-based design is limited for analogous reasons. Close inspection of the examples generated by Carroll and Rosson (1992) shows that scenarios are also device-dependent. For example, the scenario depicted in their Figure 2 (p. 187) refers to attributes of the interface (e.g., default settings in a menu). As a result, the insights that can be garnered from the analysis are limited by the interface attributes that have been specified in the scenarios.

Ideally, we would like to have a work analysis technique that makes minimal assumptions about the device (i.e., interface + automation) up front. As mentioned in chapter 3, the functionality and characteristics of the device should be an output of the work analysis. That is, the reason why we are doing an analysis in the first place is to determine how the device should be designed. The limitations of prototyping and scenarios are that they require an interface to even get started. Rather than being solely an output of the work analysis, the device specifications are needed as inputs to bootstrap the building of prototypes and scenarios. Consequently, as shown in Fig. 4.3, the task-artifact cycle is never really broken.

As for incompleteness, the insights gained from prototyping are limited by the number and range of tasks that workers are asked to perform during testing. As we discussed in chapter 3, task demands in open systems cannot be fully anticipated or enumerated in detail, so the representative set of conditions used during testing is bound to be incomplete, perhaps drastically so. Moreover, because prototyping and testing are bottom-up activities, there is no way of knowing how much of the state space of possibilities has been covered. For instance, the operators at TMI had

experienced many different situations in training and operations before the accident we described in chapter 2 occurred. However, none of those situations had been sufficient to reveal the profound deficiencies in systems design uncovered by the unique event prompting the accident. Similarly, the insights gained from scenarios are limited by the number and range of scenarios that are considered in the analysis. But as Carroll and Rosson (1992) observed: "For any reasonably complex system, the scenario representation is necessarily incomplete (there is an infinity of possible use-scenarios)" (p. 185). And again, because scenarios are generated in a bottom-up fashion by the analyst, there is no way of knowing how much of the state space of possibilities has been covered.

On the one hand, it is certainly true that breadth of coverage is likely (although not guaranteed) to improve as a function of the number of prototype and test iterations we conduct, or the number of scenarios we generate and analyze. On the other hand, incompleteness means that there will be situations for which the device has not been tailored. If these situations are encountered by workers, then the likelihood of an error is likely to be significantly greater than in the situations that have been explicitly anticipated. For work domains where the consequences of errors due to incompleteness are not severe (e.g., word processing), this situation may be acceptable. However, in many complex sociotechnical systems, the potential hazards involved are very large (see chap. 1). Thus, incompleteness becomes a much more significant concern, particularly because the accident data show that disasters occur in situations that have not been anticipated by designers.

Consequently, it is important to look for work analysis techniques that offer greater device-independence and breadth of coverage. Whatever method we choose will not offer any complete guarantees, but we must make more of an effort to overcome the limitations of descriptive work analyses and the companion techniques of prototyping and scenario building. The potential consequences of not doing so are too great to ignore, because they can be measured in enormous amounts of monetary losses, substantial damage to the environment, or tragic loss of human life.

AN ALTERNATIVE WAY OF GETTING AROUND THE TASK-ARTIFACT CYCLE: MODELING INTRINSIC WORK CONSTRAINTS

The only viable way we know of breaking free from the task-artifact cycle, and its associated limitations, is to escape from the language of tasks and artifacts. Tasks and artifacts are intimately bound to current practice, which inevitably leads to the deep-rooted problems we have discussed previously. In this section, we argue that modeling intrinsic work constraints will allow us to cut through the Gordian knot of the task-artifact cycle by identifying functional design possibilities, independently of current work practices.

Completeness: The Need for Models

As we mentioned earlier, one of the key limitations of relying solely on scenarios or on prototyping and testing is that the insights that we gain are strongly bound by the number and type of scenarios and prototypes that we can afford, or have

the ingenuity, to include. Prototype testing and scenario building are inductive, bottom-up activities and will always be incomplete because we can never test all possible prototypes under all possible conditions or analyze every possible scenario. Thus, we need to find some way of generalizing our insights to the prototypes and scenarios that we have not considered. Perhaps even more important, however, is the fact that it is not possible to determine the limits on our knowledge. With prototype testing and scenario building, we cannot systematically know what factors we have included and what factors we have missed. Each iteration is a data-driven step, and we have no knowledge of the overall space of possibilities in which we are searching.

Modeling provides a way of reducing the impact of these limitations. First, models provide an explicit, top-down basis for generalization. They generalize beyond instances by representing classes rather than exemplars. This feature of models was well captured by Ahl and Allen (1996):

> Although they are a critical part of science, data are not the purpose of science. Science is about predictability, and predictability derives from models. Data are limited to the special case of what happened when the measurements were made. Models, on the other hand, subsume data. Only through models can data be used to say what will happen again, before subsequent measurements are made. Data alone predict nothing. (p. 45)

As a result, models provide a broader basis for work analysis. Second, although they will always be incomplete, models also make explicit what attributes have been included. In fact, the process of model building is essentially that—the specification of the equivalence classes that define the model (see Glossary). Thus, models provide a more systematic and explicit basis for work analysis. Note that these features are particularly important—perhaps even essential—for complex sociotechnical systems. As Beyer and Holtzblatt (1998) pointed out, "the more complex the work, the more critical it is to maintain a coherent representation [of work]" (p. 212).

Device-Independence: Focusing on Intrinsic Work Constraints

This discussion naturally leads to the question: Model what? After all, there are many different entities that could be modeled. As this chapter has shown, the focus of attention in HCI in the past has been on artifacts and tasks. This focus leads to the problems of strong device-dependence that are inevitably caused by human adaptation. To reduce these problems, we need to find a way of escaping from the resulting regress illustrated by the task-artifact cycle in Fig. 4.3.

As shown in Fig. 4.1, one way to do this is to try to identify the set of intrinsic work constraints. By focusing on the constraints that are an inherent part of doing the job, we can try to get away from the details of how to do the job with a particular device. The primary goal in doing so is to let the work analysis suggest how the device should be designed, rather than to make assumptions about the device before the analysis even begins. By pursuing this goal, we can try to avoid including workaround activities that are a part of current practice. At the same time, we can systematically and explicitly seek out effective possibilities for accomplishing work that are not a part of current practice.

In some ways, this approach represents a radical change in work analysis and, by implication, design as well. Let us be concrete about what it really means. Work analysis methods should not prespecify:

- The existing set of sensors that are used to obtain data from the environment.
- The content and structure of the database that organizes all of the information in the information system.
- The functionality of the automation currently in place.
- The allocation of functions between computers and workers.
- The allocation of job responsibilities to individuals or groups.
- The appearance and structure of the interface.
- Workers' competencies.

The reason why work analysis should not inherit these decisions is because each of these issues is a point of design leverage. As a result, they should not be inputs into the work analysis, otherwise we would be inheriting the vestiges of the old, and likely inadequate, design. Instead, the decisions concerning what information should be gathered, how it should be organized, how to automate, what to automate, how to organize work, how to display information, and how to train operators should all be made based on the findings (i.e., the outputs) obtained from the work analysis. The very point of conducting a work analysis is to provide an informed basis for making decisions about these points of design leverage. To adopt the existing, or a priori, solutions to these issues is to miss important opportunities for creating novel and more effective ways of supporting human work. In the words of Beyer and Holtzblatt (1998), this approach "makes deciding how customers will work in the future the core design problem and uses those decisions to drive the use of technology" (p. 3).

As we have pointed out repeatedly in this chapter, this is not to say that it is not valuable to study current work practices. On the contrary, as the overlap in the center of Fig. 4.1 shows, studying current practice can shed a great deal of light on intrinsic work constraints. However, the analysis of current practice should be viewed as one of several possible means to investigate work constraints, rather than an end in itself. It is the work constraints themselves that should take precedence, if the pitfalls identified earlier are to be avoided.

There are several other ways of identifying intrinsic work constraints. For example, workaround activities are frequently indirect pointers to functionality that could be supported by the device (e.g., Vicente, Burns et al., 1996). In addition, sometimes workers try to use certain strategies but fail to carry them out effectively, not because the strategies are not useful but because the support to implement them reliably is not currently available. These strategies represent currently unexplored possibilities for doing work (see chap. 9). Also, by studying the structure of the work domain, it may be possible to identify tasks that should be performed but that are not currently being performed. Similarly, by studying the structure of the work domain and the tasks that need to be performed, it may be possible to identify novel strategies that could be used by workers. Finally, in some application domains, it may be possible to use analytical models (e.g., based on operations research techniques) to identify potentially very effective but currently unexplored possibilities for accomplishing work (e.g., Dessouky, Moray, & Kijowski, 1995).

SUMMARY

What type of work analysis is appropriate for complex sociotechnical systems? In this chapter, we evaluated the suitability of descriptive approaches in the form of field studies of current practice in naturalistic settings. Such studies have convincingly shown the strong limitations of normative approaches to work analysis discussed in chapter 3. Descriptive approaches have also led to important implications for work analysis. In addition to the work domain and constraint-based task analyses reviewed in the previous chapter, a framework for work analysis must also address strategies, social-organizational factors, and competencies for expertise.

Despite these valuable insights, descriptive approaches have limitations of their own, stemming from the regress captured by the task-artifact cycle. To bootstrap a descriptive work analysis, an existing, simulated, or envisioned device is required. As a result, descriptive approaches inherit the deficiencies of current practice, namely workarounds and currently unexplored possibilities (see Fig. 4.1). Thus, it is not generally useful to design future devices strictly to support current work practices.

Proponents of descriptive approaches to work analysis have proposed several techniques to get around the task-artifact cycle, the most prominent being prototypes and scenarios. Although certainly very useful—and perhaps even sufficient for some types of design problems—these techniques are not sufficient to satisfy fully the needs of complex sociotechnical systems. Prototypes and scenarios are both limited by problems of strong device-dependence and incompleteness. These limitations can be directly addressed by explicitly modeling intrinsic work constraints. With this approach, the various characteristics of the device and the organizational structure of work emerge as outputs of the work analysis rather than being required as inputs to get the analysis started. The bottom line is that descriptive studies of current practice can provide a useful window into intrinsic work constraints, but a different perspective is required to cut through the task-artifact cycle. In short, a formative approach to work analysis is needed to deal with the unique needs of complex sociotechnical systems.