# 3 Scenario-Based Design

Designers of information systems and applications face the challenges all other designers face. The problems they address become definitively analyzed only by being solved; appropriate solution methods are never givens, but are confirmed by being tried out; solutions must be implemented in order to be specified. These designers face convoluted networks of trade-off and interdependency, the need to coordinate and integrate the contributions of many kinds of experts, and the potential of unintended and untoward impacts on people and their social institutions. Beyond all this, they face the likelihood that changing cultural and technological circumstances will obviate any solution they develop before it can be deployed.

Most engineering methods belong to the methodological tradition that seeks to *control* the complexity and fluidity of design through techniques that filter the information considered and decompose the problems to be solved. Scenario-based design techniques belong to a complementary tradition that seeks to *exploit* the complexity and fluidity of design by trying to learn more about the structure and dynamics of the problem domain, trying to see the situation in many different ways, and interacting intimately with the concrete elements of the situation.

In scenario-based design, descriptions of how people accomplish tasks are a primary working design representation. Software design is fundamentally about envisioning and facilitating new ways of doing things and new things to do. Maintaining a continuous focus on situations of and consequences for human work and activity promotes learning about the structure and dynamics of problem domains, seeing usage situations from different perspectives, and managing trade-offs to reach usable and effective design outcomes (Carroll 1994, 1995).

## What Are Scenarios?

Computers are more than just functionality. They unavoidably restructure human activities, creating new possibilities as well as new difficulties. Conversely, each context in which humans experience and act provides detailed constraint for the development and application of information technologies. In analyzing and designing systems and software, we need better means to talk about how they may transform and be constrained by the contexts of human activity. This is the only way we can hope to attain control over the "materials" of design. A direct approach is explicitly to envision and document typical and significant activities early and continuingly in the development process. Such descriptions, often called scenarios, support reasoning about situations of use, even before those situations are actually created.

Scenarios are stories—stories about people and their activities. For example, an accountant wishes to open a folder on the system desktop in order to access a memo on budgets. However, the folder is covered up by a budget spreadsheet that the accountant wishes to refer to while reading the memo. The spreadsheet is so large that it nearly fills the display. The accountant pauses for several seconds, resizes the spreadsheet, moves it partially out of the display, opens the folder, opens the memo, resizes and repositions the memo, and continues working. This is about as mundane a work scenario as one could imagine. Yet it specifies window management and application switching functionality vividly and pointedly: people need to coordinate information sources—to compare, copy, and integrate data from multiple applications; displays inevitably get cluttered; people need to find and rearrange windows in the display.

Scenarios highlight goals suggested by the appearance and behavior of the system; what people try to do with the system; what procedures are adopted, not adopted, carried out successfully or erroneously; and what interpretations people make of what happens to them. If the accountant scenario is typical of what people want to do, it substantively constrains design approaches to window management and switching.

Scenarios have characteristic elements (Propp 1958). They mention or presuppose a *setting*. For the accountant scenario, the setting is an office, with a person seated in front of a computer. The setting includes the relative positions of the folder and spreadsheet. The scenario implies further setting elements by identifying the person as an accountant, and the work objects as budgets and memos.

Scenarios include *agents* or *actors*. The accountant is the only agent in this example, but it is typical of human activities to include several to many agents, each typically with *goals* or *objectives*. These are changes that the agent wishes to achieve in the circumstances of the setting. Every scenario involves at least one agent and at least one goal. When more than one agent or goal is involved, they may be differentially prominent in the scenario. Often one goal is the defining goal of a scenario—the answer to the question, "Why did this story happen?" Similarly, one agent might be the principal actor—the answer to the question, "Who is this story about?" In the accountant scenario, the defining goal is displaying the memo in such a way that both the memo and budget can be examined. A subgoal is opening the folder in which the memo is located, and a further subgoal is moving the budget to allow the folder to be opened.

Scenarios have a plot; they include sequences of *actions* and *events*, things that actors do, things that happen to them, changes in the circumstances of the setting, and so forth. Particular actions and events can facilitate, obstruct, or be irrelevant to given goals. Resizing the spreadsheet and moving it out of the display are actions that facilitate the goal of opening the folder. Resizing and repositioning the memo are actions that facilitate the goal of displaying the memo so that it can be examined with the budget. Pausing is an action that is irrelevant to any goal, though it suggests that the accountant's goal-oriented actions were not completely fluent. Notably, actions and events can often *change* the goals—even the defining goal—of a scenario.

Representing the use of a system or application with a set of interaction scenarios makes that *use* explicit, and in doing so orients design and analysis toward a broader view of computers. It can help designers and analysts to focus attention on the assumptions about people and their tasks that are implicit in systems and applications.

Scenario representations can be elaborated as prototypes, through the use of storyboard, video, and rapid prototyping tools. They are the minimal contexts for developing use-oriented design rationale: a given design decision can be evaluated and documented in terms of its specific consequences within particular scenarios. Scenarios and the elements of scenario-based design rationale can be generalized and abstracted using theories of human activity, enabling the cumulation and development of knowledge attained in the course of design. Thus, scenarios can provide a framework for a design-based science of human-computer interaction.

By making *use* the object of design, scenarios can be tools to address the fundamental issues about design raised in chapter 2. The balance of this chapter reviews some key challenges for design work and illustrates for each the benefits of a scenario-based approach.

## Challenge: Design Action Competes with Reflection

There is a fundamental tension between thinking and doing: thinking impedes progress in doing, and doing obstructs thinking. Information technology designers are skilled practitioners performing complex and open-ended tasks. They want to reflect on their activities, and they routinely do reflect on their activities. However, people take pride not only in what they know and learn, but in what they can do and in what they actually produce. They know from experience that it is impossible to fathom all potential consequences and interdependencies, and they can be frustrated by problem clarification and discussion of alternatives. They want to act—to make decisions and see progress.

In the multimedia education project, the design team was tackling a large and ill-defined problem. The team included a variety of specialized expertise in natural language processing, multimedia authoring, and user interface design, and had broad goals and interests. At the start of their work, they held regular and lengthy meetings to develop a singular vision and a comprehensive plan. They seemed to enjoy the early brainstorming meetings, but increasingly felt it was time to get going, although many issues remained open. They shifted into action mode at the expense of reflection and coordination; the weekly team meetings were quietly canceled. In the end several groups within the project pursued poorly integrated subgoals.

Donald Schön (1983 1987) has discussed this conflict extensively in his books on reflective practice. For example, he analyzes a coaching interaction between two architects in which the student presents a design concept for a school building involving a spiral ramp, intended to maintain openness while breaking up lines of sight (she calls the idea "a Guggenheim"): "When I visited open schools, the one thing they complained about was the warehouse quality—of being able to see for miles. It [the ramp] would visually and acoustically break up the volume" (1983, p. 129). The coach feels that she needs to explore and develop her concept more thoroughly, noting that a ramp has thickness and that this will limit her plans to use the space underneath the ramp. He urges her to draw sections but without bothering to justify this advice to the student, to allow her to see the rationale behind his advice. As Schön (1987) puts it, he does not reveal "the meanings underlying his questions" (p. 132).

Schön regards this as a hopeless confrontation in which no progress can be made on the particular design project or the larger project of understanding how to design. Both the student and the coach are willing to act publicly and to share actions, but neither adequately reflects on their own or one another's values and objectives. Reflection is not always comfortable; it forces one to consider one's own competence, to open oneself to the possibility of being wrong. As Schön (1987) put it, "The interactions I have suggested emphasize surfacing private attributions for public testing, giving directly observable data for one's judgments, revealing the private dilemmas with which one is grappling, actively exploring the other's meaning, and inviting the other's confrontation of one's own" (p. 141).

Reflective activities in information system design often occur decoupled from action. Design review meetings allow a project team to take stock of progress by working through a set of requirements, a progress report, a specification, or to exercise a prototype. Such reviews can facilitate design work in many ways, clarifying problems, moves, goals, trade-offs, and so forth. However, design reviews remove designers from the day-to-day context of their work to facilitate reflection. They provide an opportunity to stop design work as such and reflect on interim results. Reviews do not evoke reflection *in the context of doing design.* Moreover, design reviews typically focus on enumerating and defining features and functions, as opposed to reviewing the potential use of a system.

**Box 3.1**
Scenario of use for a multimedia educational system

> Harry is interested in bridge failures; as a child, he saw a small bridge collapse when its footings were undermined after a heavy rainfall. He opens the case study of the Tacoma Narrows Bridge and requests to see the film of its collapse. He is stunned to see the bridge first sway, then ripple, and ultimately lurch apart. He quickly replays the film, and then selects the associated course module on harmonic motion. He browses the material (without doing the exercises), saves the film clip in his workbook with a speech annotation, and then enters a natural language query to find pointers to other physical manifestations of harmonic motion. He moves on to a case study involving flutes and piccolos.

Evaluation is another important reflective development activity, and one that often does address use directly, but evaluation tends to occur late in the system development process, and thus also is largely decoupled from design action. Schön's notion of reflection-in-action is an approach to design activity itself, as contrasted with iterative feedback cycles of design activity followed by reflective analysis in design reviews or evaluation.

## Scenarios Evoke Reflection in Design

Constructing scenarios of use as focal design objects inescapably evokes reflection in the context of doing design. The scenario in box 3.1 succinctly and concretely conveys a vision of the system—in this case a vision of student-directed, multimedia instruction. It is a coherent and concrete vision—not an abstract goal, not a set of requirements, not a list of features and functions. Elements of the envisioned system appear in the scenario embedded in the interactions that will make them meaningful to people using the system to achieve real goals—perhaps revelatory, perhaps cryptic, but definitely more than just technological capabilities. For example, the role of natural language query is exemplified as a means of locating further case studies that illustrate the principles of harmonic motion.

The scenario emphasizes and explores goals that a person might adopt and pursue, such as watching the film clips twice or skipping the exercises. Some of these goals are opportunistic, such as investigating the Tacoma Narrows Bridge collapse because of experience with a totally unrelated bridge collapse or deciding to branch from bridge failures to flutes. The scenario implicitly articulates the usage situation from multiple perspectives: the student stores and annotates a video clip with speech, raising specific requirements for user interface tools and presentation as well as for particular data structures and memory. The scenario impels the designer to integrate the consideration of such system requirements with consideration of the motivational and cognitive issues in education that underlie the person's actions and experiences.

Schön (1983) drew an important contrast between merely creating or identifying elements of the problem context and "giving them reason." He characterized design as inquiry and the designer as "a researcher in a practice context" (p. 68). From this perspective, the designer's domain expertise is both a prerequisite and a potential liability. Having a category or a label with which to "understand" a situation often establishes an insurmountable obstacle to achieving an innovative understanding. In the Guggenheim example, the student sees the design only as a geometry affording patterns of light and sound. However, in her sketches, the Guggenheim analogy is static; it is a design answer that fails to evoke critical questions about the space underneath the ramp.

Creating and using scenarios pushes designers beyond static answers. The Guggenheim is an environment for walking and sitting underneath the ramp, among other activities. Embodying the activities of the Guggenheim space instantly raises questions about the space. The scenario in box 3.1 conveys the functionality of the system through vivid details about how a person will access that functionality and what a person might experience in doing so. This emphasis on raising questions makes it easier for designers to integrate reflection and action in their own design practice.

In strategic management, scenarios have played this sort of role. For example, Wack (1985a, 1985b) describes the use of scenario-based planning in Royal-Dutch Shell in the early 1970s. By developing a set of future scenarios, Shell was able to anticipate the emergence of OPEC, including the oil crisis of the 1973 and the oil glut of the early 1980s. Wack (1985a) concluded that the use of scenarios allowed Shell managers to raise questions, challenge assumptions, and understand the dynamics of the business environment “more intimately than they would in the traditional planning process” (p. 74).

In human-computer interaction, scenarios have been used in many system development activities. They are widely used for planning and describing evaluation test tasks (Roberts and Moran 1983) and task-oriented instruction and other user support (Carroll 1990a). They are used to specify usability goals in design (Carroll and Rosson 1985), and to frame design rationale (Carroll, Kellogg, and Rosson 1991) and design reviews (Karat and Bennett 1991). However, most of these uses are pursued in isolation from one another and from *acts* of design. That is, most uses of scenarios in human-computer interaction are not coordinated with one another and occur prior to or after design itself.

Schön (1983) emphasized that effective reflection and action must be tightly coupled. A design analysis need not be comprehensive; it need only guide a restructuring of the current situation that can produce new design actions and new insights. Schön contrasted such interactive and concrete “move testing” activity with traditional hypothesis testing in which abstractions are decomposed, instantiated, evaluated, and then either rejected or maintained. His cases are drawn from design domains for which there are rich languages to allow practitioners to create and explore situations quickly, for example, by sketching in architecture. Wack (1985b) also makes the point that effective scenarios must evoke planning and redesign action.

Scenarios provide such a language for the design of human-computer interactions. Kahn (1962), who used scenarios to analyze cold war dynamics, compared the role of scenarios in understanding human action to that of theories of mass and velocity in physics. They allow analysts and designers to sketch interactions in order to probe relationships; they resolve the paradox of reflection and action by providing a language for action that is ineluctably reflection evoking.

## Challenge: Design Situations Are Fluid

Design changes the world within which people act and experience, and this changes requirements for further designs. When designs incorporate new technologies and address new arenas of human activity, such as electronic meetings or desktop manufacturing, requirements evolve more rapidly. New design moves and new design goals become possible and necessary to address these requirements. In such cases, there is little technological or social inertia to impede change, and little knowledge or experience to provide overall guidance. Design problems in information technology often change significantly during the course of their own solution process.

The fluidity of information system design is more than merely a source of technical complexity. Schön (1967) observed that in our era, technology undermines the stability of the world. For example, it erodes a manager's concept of "the business I'm in" (p. 195) and a worker's notion of the special skill he or she possesses; it erodes the traditional staging of life into education followed by steady practice. In everyday life, people create illusions of stability to manage their own uncertainty about situations, even if they must adjust their perceptions in order to do this (Nisbett and Wilson 1977). They protect their interpretations by selectively reconstructing their own experience (Erikson 1980). Interestingly, when greater reconstructive effort is required to maintain an interpretation, it is held all the more ardently, even in the face of incontestable discomfirmation (Festinger 1957; Festinger, Riecken, and Schachter 1956). Thus, a typical response to ambiguities in life is to fix interpretations and then defend them as long as possible.

The analogous response to the fluidity of design situations is top-down design and systematic decomposition. Designers fix a top-level concept based on their initial understanding of a design problem. They decompose this overall concept into concrete, enabling subgoals, as in Simon's house. If the top-level concept subsequently turns out to have been a favorable choice, this is called inspiration. However, this approach trades flexibility to avoid uncertainty; the top-level concept will often turn out to have been a premature commitment, painting the design into a corner from which it cannot address requirements that emerge through the course of the design process. Schön (1967) observed that decomposition is useful chiefly through the *side-effects* of eliciting and focusing concrete action, and thereby reducing uncertainty. But he warned that this approach frequently strangles innovation.

Designers almost always design something. If their need for stability induces them to close their designs prematurely, to cease reflecting and critiquing, to declare victory, they may produce a solution for only the first few requirements they encounter. However, struggling against premature commitment is difficult because it is such a general psychological strategy for managing uncertainty in life.

## Scenarios Are at Once Concrete and Flexible

To manage an ambiguous and dynamic situation, one must be concrete but flexible—concrete to avoid being swallowed by the indeterminacies, flexible to avoid being captured by a false step. Scenarios of use reconcile concreteness and flexibility. They are concrete in the sense that they simultaneously fix an interpretation of the design situation and offer a specific solution. The scenario in box 3.1 specifies a particular usage experience that could be prototyped and tested. At the same time, scenarios are flexible in the sense that they are deliberately incomplete and easily revised or elaborated. In a few minutes, a piece of the scenario could be rewritten (for example, stipulating a system-initiated prompt for the associated course module on harmonic motion) or extended (for example, the objects in the scenario could be described as anchors for a variety of link types).

Scenarios embody concrete design actions and evoke concrete move testing on the part of designers. They allow designers to try things out and get directive feedback. But their flexibility facilitates innovative and open-ended exploration of design requirements and possibilities, helping designers to avoid premature commitment. This allows designers to manage the three difficult properties of design that Reitman (1965) emphasized: incomplete problem requirements, little guidance on possible methods, and open-ended goal states. Scenarios are Dreyfuss's clay mock-ups for the domain of interaction design.

The power of scenarios to convey concreteness through tentative and sketchy descriptions derives from the way people create and understand stories. Like the strokes of an expressionist painting, scenarios can evoke much more than they literally present. The human mind seems especially adept at overloading meaning in narrative structures, in both generation and interpretation, as illustrated by the remarkable examples of dreams (Freud 1900) and myths (Lévi-Strauss 1967). Indeed, dreams and myths are universal tools for coping with uncertainty. Part of their power stems from being recalled and discussed throughout a community. In a more modest scale, design teams can do this too. Sharing and developing scenarios helps to control the uncertainties inherent in the work, while strengthening the team itself.

Indeed, Simpson (1992) identified overspecification of scenarios as a potential pitfall. And several writers have emphasized the importance of descriptive and memorable names for scenarios (Schoemaker 1995; Simpson 1992; Wack 1985a). Once the scenario has been developed, a vivid name is often enough detail. In strategic management, scenario-based planning and analysis has been found to be most useful when uncertainty is high, there are strong differences of opinion about courses of action, and communication is poor (Schoemaker 1995).

Managing complex and fluid problem spaces with concrete scenarios is a natural approach. Designers already do it—both the library and multimedia system projects used scenarios. However, the role that scenarios play in design is often ad hoc, casual, fortuitous. Even when they are used deliberately, scenarios are rarely used systematically or comprehensively to support coordinated reflection and action. Yet it is a small step from current practice to exploit more thoroughly the possibility of keeping design commitments provisional, and thereby evoking further innovative ideas, through deliberate sketchiness.

## Challenge: External Factors Constrain Design

Designers must have constraints; there are just too many things that might be designed. Requirements, if they can be identified accurately, are clearly the best source of constraints because they indicate what sort of design work is needed. But there are many other sources of constraint external to the design problem. The current state of technology development, for example, makes some solutions impossible and others irresistible. Designers cannot use technology that does not yet exist, though their work often drives technology development toward possibilities that are barely within reach. Conversely, designers and their clients are caught up in the technological zeitgeist, biased toward making use of the latest gadgets and gizmos, perhaps even when these novelties are not really appropriate to serve the user's needs. Finally, designers are biased toward employing technologies they have used before and are part of their personal design palette. Technological possibilities are a sort of window within which design work occurs.

Many external constraints in design originate in the organizational structures within which design work is embedded. Development organizations often systematically confuse clients with users, taking client concerns as user requirements. Certain types of assumptions and arguments may be favored in the business cases that underwrite design projects. For example, the multimedia project overemphasized technological objectives and constraints, uncritically assuming that students and professors would rather send video messages to one another than meet face to face. In the 1970s, it was often asserted that executives would never use a keyboard, and this became a working assumption. Such underlying attitudes restrict the designer's opportunity to understand and respond to the actual problem situation.

Organizations also specify work procedures that constrain designers. In information system development, it is still typical for design teams to work within some sort of waterfall framework, handing off documents and partial results to other groups. It is difficult to ensure adequate coordination and transmission of key information in such processes (Poltrock and Grudin 1994). Indeed, procedures involving hand-offs of responsibility and control often make it *more* difficult for people with different kinds of expertise to work together effectively. For example, it is not uncommon for "user interface" and/or "usability" to be broken out as separate subprocesses in a waterfall; this has the effect of removing or subordinating interaction design with respect to the design of system functionality. Since it is common for new requirements, methods, and goals to emerge in later stages of system development, waterfall procedures typically entrain iterations of reworking. Management structures are not part of the design problem, but they constrain how designers can work on problems.

Further external constraints arise in deploying new technology. The people who commission or select new technology are frequently not the ones who will use it, and the people who acquire and use new technology are not the only ones affected by its use. In the simple case, the technology may be difficult to use or just ineffective in the situations for which it has been acquired. But there are many further possibilities. Other tasks and activities that the users engage in, or the tasks and activities of other people in the organization with whom the users interact, may be detrimentally affected by the technology. The technology may indirectly exacerbate or expose problems in the organization itself; it may even become a lightning rod for organizational disputes that have nothing to do with any relevant task or activity.

External constraints are the larger context of design and use. Designers must cope with the current state of technology, their own work organization, and the wide-ranging and continuing impacts their work can have on client organizations. The key risk posed by external constraints is that they can distract designers from what is essential and tractable in the design project: addressing the internal constraints of the problem, the needs and concerns of the people who will use and experience the design result.

## Scenarios Promote Work Orientation

Scenarios are work-oriented design objects. They describe systems in terms of the work that people will try to do as they make use of those systems. A design process in which scenarios are employed as a focal representation will ipso facto remain focused on the needs and concerns of prospective users. Thus, designers and clients are less likely to be captured by inappropriate gadgets and gizmos, or to settle for routine technological solutions, when their discussions and visions are couched in the language of scenarios, that is, less likely than they might be if their design work is couched in the language of functional specifications (Carroll and Rosson 1990).

Kahn (1962) emphasized the utility of deliberately envisioning and analyzing extreme cases in order to stimulate broader imagination of a range of future possibilities. Scenario-based analysis helps organizations to steer between the twin risks of overconfidence—attempting to change or accomplish too much, and conservatism—attempting to do too little (Clemons 1995).

Scenarios can help to integrate the variety of skills and experience required in a design project by making it easier for different kinds of experts to communicate and collaborate (Clausen 1993). Among the developers, scenarios can ease problems associated with handoffs and coordination in the design process by providing all groups with a guiding vision of the project goal to unify and contextualize the documents and partial results that are passed back and forth. Such a systematic use of scenarios was notably lacking in the multimedia education project.

All stakeholders speak the language of scenarios. Thus, employing scenarios as a focal design representation facilitates participatory design (Kyng 1995; Muller et al. 1995). But beyond merely allowing users and other clients to participate, scenarios help to keep technical considerations regarding use first and foremost in the face of potentially distracting considerations stemming from the organizations of the developers and the users. Supporting direct participation of users and other client stakeholders is a proactive approach to untoward organizational impacts that may obtain when the technology is inserted into the work context.

Wack (1985a) reported that a key to the Shell approach was directly involving management in the scenario work. Simpson (1992) found that using scenarios helped to clarify management assumptions in a planning process. Stokke et al. (1991) used scenario-based planning to design a long-range research and development strategy for Norway's state-owned oil and gas company, reporting that the methodology was simple and transparent, the process was highly flexible, scenarios helped keep the stakeholders focused on relevant issues, and that there was a high degree of ownership in the final product. They concluded that adopting a scenario-based method started to transform their company into a learning organization.

## Challenge: Design Moves Have Many Consequences

Schön (1983) sees design as a "conversation" with a situation comprising many interdependent elements. The designer makes moves and then "listens" to the design situation to understand their consequences: "In the designer's conversation with the materials of his design, he can never make a move which has only the effects intended for it. His materials are continually talking back to him, causing him to apprehend unanticipated problems and potentials." Thus, the systems engineer and the librarian made the move of spooling print jobs and then noticed the (apparently) unanticipated potential of redistributing the entire on-line reference system. When a move produces unexpected consequences, and particularly when it produces undesirable consequences, the designer articulates "the theory implicit in the move, criticizes it, restructures it, and tests the new theory by inventing a move consistent with it" (Schön 1983, p. 155).

A classic design rationale method for representing design conversations is the issue-based information system (IBIS) of Rittel and Weber (1973). This approach describes the design process as a graph of issues arising, each pointing to positions, or alternatives, enumerated as responses, each of which in turn points to arguments for and against that position. In this approach, it is relatively easy to trace single-issue threads, but more difficult to understand why a given alternative (that is, a design element) makes sense in the context of the many issues it may simultaneously address, and the many arguments that bear on it with respect to those issues.

Every element of a design, every move that a designer makes, has a variety of potential consequences. Spooling the local printer output downstairs to the mainframe computing facility had the direct consequence of freeing some floor space near the terminals in the library reference-circulation area. And it also integrated the mainframe facility into the library's on-line reference system, albeit quite loosely. This integration, in turn, suggested the possibility of a more central role in which the mainframe would act as the reference system client for the whole research center.

This design had wide-ranging impacts on the software for accessing on-line references, on the use of library and reference material, and indeed on work responsibilities and social interactions through the research center. The new design distributed access to on-line reference databases through the center. This entailed that the computing center's mainframes, and not the library's terminals, would manage access to these databases. This reallocated work among several groups in the research center staff: the reference librarians ended up with fewer face-to-face reference searches, but the computing center staff ended up with a new function to support. And it affected a myriad of other tangential interactions and activities—for example, carrying out reference searches from one's own office diminishes opportunities for chance encounters with colleagues in the library.

Many of the most common planning techniques are ill suited for problems in which many uncertainties must be simultaneously understood and managed (Schoemaker 1995). Contingency planning examines the impact of one binary uncertainty at a time—for example, what options do we have if we can spool printer output? Sensitivity analysis examines effects of parametrically changing one variable at a time, keeping all others constant—for example, how many additional local print jobs can we expect if the research staff increases by 5 percent? Simulation models make contingency planning and sensitivity analysis more tractable, but the downside they entail is a potentially huge outcome space with little semantic structure.

Designers need a language for keeping track of the conversation with the design situation, for recognizing and addressing the numerous tradeoffs and dependencies among elements of the design problem. They need techniques for comprehensively managing of design moves, consequences for software, for directly supported tasks and activities, for work organizations, and for incidental social interactions.

## Scenarios Have Many Views

Scenarios of use are multifarious design objects. They allow designers to explore joint impacts of sets of uncertainties. They support changing the values of several key variables at a time to describe new states that may arise after major shocks or deviations in key variables. They give interpretations to patterns or clusters of possible outcomes, concretely structuring the outcome space. And they describe designs at multiple levels of detail and with respect to multiple perspectives.

A scenario can briefly sketch tasks without committing to details of precisely how the tasks will be carried out or how the system will enable the functionality for those tasks. The multimedia education scenario in box 3.1 is at an intermediate level, with some detail regarding task flow, but it could be elaborated with respect to Harry's moment-to-moment thoughts and experiences in order to provide a more elaborated cognitive view, or with respect to individual interactions to provide a more detailed operational view. Alternatively, the scenario could be presented from the point of view of someone watching Harry, perhaps in order to learn how to operate the system. It could be elaborated in terms of Harry's organization, emphasizing the broader context of his activity—the people with whom he works and the goals they are jointly pursuing. It could be elaborated in terms of hardware and software components that could implement the envisioned functionality in order to provide a system view. Each of these variations in resolution and perspective is a permutation of a single underlying use scenario. Indeed, the permutations are integrated through their roles as complementary views of the same design object.

Scenarios can leave implicit the underlying causal relationships among the entities in a situation of use. For example, in box 3.1 the envisioned speech annotation capability allows adding a personal comment without the overheads of opening an editor and typing text. However, the annotation is noncoded and thus cannot be edited symbolically. These relationships are important to the scenario, but often it is enough to imply them. Similarly, scenarios can explicitly detail temporal relations, as in a Gantt chart, or they can leave sequence and duration vague, as in box 3.1. This is an aspect of the property of being concrete but rough.

Sometimes, however, it can be useful to make these relationships explicit. For example, in another scenario of use, Harry may wish to collect and revisit the set of film clips he viewed and annotated as "breakthroughs in forensic engineering." Unfortunately, his noncoded voice annotations cannot be searched by string. Thus, this new scenario would end in failure. To understand and address the variety of desirable and undesirable consequences of the original annotation design move, the designer might want to make explicit the relevant causal relationships in these two scenarios, thus providing yet another view of the envisioned situations, as shown in box 3.2.

Scenarios can help guide the designer's move toward consequences of differing types. For example, the data structures for the workbook might differentiate between annotated and nonannotated items, allowing annotated items to be retrieved and browsed as a subset. This would not allow Harry to retrieve the set of items with a particular annotation directly, but

**Box 3.2**
View of the multimedia education scenario sets of consequences associated with the orientational video clip and the speech annotation capability

A video clip of the Tacoma Narrows Bridge collapse
+ provides an engaging introduction to a course module on harmonic motion
+ evokes curiosity and self-initiated learner exploration
– may not sufficiently motivate a learner to attempt the course module

Speech annotation of a saved video clip
+ allows easy attachment of personal metadata
– does not support indexing or search of metadata

it would still simplify the search. Alternatively, the search problem might be addressed directly by speech recognition or audio matching, or by including the option of text annotation. Each of these alternatives would entrain different elaborations for both the annotation scenario in box 3.1 and the search scenario just discussed. These elaborations could then be explored for further consequences and interdependencies.

Thus, one would want to pursue the usability consequences of the various elaborations, for example, the frustrations of a 90 percent recognition accuracy. One would want to investigate trade-offs and interdependencies among consequences of different types. The speech recognition capability might resolve the search scenario, but it might entail prohibitive costs in memory and processing. Including an option of text annotation might change the annotation task in a subtle way; the person would be aware when creating the annotations that they will later be retrieval keys. Providing a finely articulated data structure for the workbook enables flexible organization and retrieval, but at the cost of complexity in the underlying database.

Using scenarios in this way makes them an extremely powerful design representation. They allow the designer the flexibility to develop some use scenarios in great detail—for example, the ones that describe the core application functionality or the critical usability challenges, while merely sketching other less problematic scenarios. At the same time, they allow the designer to switch among scenario perspectives and directly integrate, for example, usability views with system and software views. Such a flexible and integrative design object is precisely what designers need to manage the nexus of consequences entrained by their design moves.

## Challenge: Technical Knowledge Lags Design

Designers often work in domains for which certain critical scientific and engineering knowledge is missing. This is typical in the design of information technology. The multimedia project went forward without good theories about how people work together in networked multimedia environments in general or in educational contexts specifically. There was little science—indeed, little experience at all with such systems to draw on.

Schön (1983) emphasized the "dilemma" of rigor or relevance throughout the professions. He describes the "high, hard ground" where practitioners can make use of systematic methods and scientific theories, but can address only problems of relatively limited importance to clients or to society. He contrasts this to the "swampy lowland [of] situations . . . incapable of technical solution" (p. 42). Here the practitioner can confront the greatest human concerns, but only by compromising on technical rigor. The design and development of technology aspires to occupy the high, hard ground, to apply the current state of science and engineering knowledge systematically, reduce difficult problems to mere corollaries—and then bask in the confidence that deductive science confers on both its practitioners and the recipients of their efforts: science is certainty. But at the same time, technology design and development is ineluctably driven to pursue novelty and innovation, and thereby to slip continuingly into the swampy unknown regions.

It can resist this only through self-destructive willfulness. During World War II, operations research developed from the successful application of mathematics to modeling battle situations like submarine search. After the war, the field expanded its interests to management problems in general, and successfully produced effective formal models for relatively bounded problems like managing capital equipment replacement or investment portfolios. However, it generally failed in complex, less well-defined areas like business management, housing policy, and criminal justice. Remarkably, the response of the field was largely to focus attention on the relatively simple problems that suited the mathematical models (Schön 1983).

Russell Ackoff (1979), one of the founders of operations research, lamented the inability and disinterest of the field in addressing the "turbulent environments" of the real world: "Managers are not confronted with problems that are independent of each other, but with dynamic situations that consist of complex systems of changing problems that interact with each other. I call such situations *messes*. Problems are abstractions extracted from messes by analysis; they are to messes as atoms are to tables and charts . . . Managers do not solve problems: they manage messes" (pp. 99–100). This quotation evokes the story in which a man loses his car keys one night, but decides to search for them not where they were dropped but under a nearby streetlight *because the light is so much better.* The man will not find his keys, and Ackoff analogously worried that solving operations research problems will not lead to "designing a desirable future and inventing ways of bringing it about" (p. 100).

The challenge is to build hard ground in the swamp of real design problems. If we never build abstract and generalizable theories of design domains, every design project must start from scratch. A beacon for this is the Catgut Society's research on the physics of violins (Hutchins 1981). At the start of this project, there was good science for analyzing a single vibrating string, but nothing that could analyze vibrating systems composed of strings spanning air cavities partitioned by wooden ribs and bounded by irregularly shaped wooden boxes. Integrating acoustical research with the design of new stringed instruments, the Catgut group scaled the acoustics of the violin to obtain a complete set of eight acoustically balanced instruments, addressing the traditional problems of the lack of a tenor voice among string instruments and the unbalanced sound power of violas and cellos. Through this design work, the Catgut group also created new fundamental knowledge in physics.

## Scenarios Can Be Abstracted and Categorized

Although technical design cannot escape Schön's swamp, designers need to extract lessons from their experience to guide their work and improve their practice. If scientific knowledge lags design, then designers themselves must formulate what they learn. Alexander and his collaborators (1977) pioneered this sort of approach to design knowledge in their concept of pattern. This takes the notion of design as inquiry quite seriously and raises the question of what abstractions can support the development and reuse of knowledge in design.

Scenarios keep the designer of information systems and applications in the swamp, connected to the turbulent real world, but by their very nature, scenarios also provide scaffolding to get a view of the design situation from a bit higher up. The roughness of scenarios is, after all, a homey kind of abstractness. A scenario is a "middle-level" abstraction (Mills 1959) of a set of possible interactions. For example, the scenario in box 3.1 could guide the design of a multitude of information systems. Documenting and explaining a scenario provides an account of a class of possible situations; this creates a generalized design object that can be immediately employed in further design work by being elaborated in various ways. For example, the scenario could be developed as a "protective" scenario, anticipating risks associated with decisions, or as an "entrepreneurial" scenario, codifying new strategic opportunities (Wack 1985b).

But this is only the simplest sense in which scenarios can be used to develop design knowledge. Scenarios exemplify particular themes and concerns in work and activity situations. Thus, Schwartz (1992) describes three main "plots" for strategic management scenarios: winners and losers, challenge and response, and evolution. For example, winners and losers plots assume a zero-sum game for resources or success. He recommends that each of these main plots be instantiated in projecting possible futures in organizational planning and that analysis of given scenarios always include a diagnosis or declaration of the scenario type.

In one of the scenarios for a multimedia education system discussed earlier (see box 3.1), a person is at first opportunistically exploring an information structure, but eventually adopts a particular interest that guides his exploration. In another scenario, a person wishes to search and organize information that has previously been browsed. Described at this level of generality, these are not scenarios unique to multimedia education systems or even to computers. They are general patterns for how people work with information. Therefore, it is likely that some of the lessons learned in managing the “opportunistic exploration” pattern or the “searching under a description” pattern in the design of any given situation might be applicable to the subsequent design of other situations. Such a taxonomy of scenarios would provide a framework for developing technical design knowledge.

Scenarios can also be classified in terms of the causal relations they comprise. In box 3.1, for example, providing speech annotation simplifies the actions needed to personalize a piece of information. In this causal relation, the consequence is the simplification of organizing and categorizing, a general desideratum in designing interactive systems. Generalizing the relation in this way allows the feature associated with the consequence (in this example, speech annotation) to be understood as a potential means for that consequence and employed to that end in other design contexts. There is, of course, no guarantee that the generalization is correct; that can be settled only by trying to use it and succeeding or failing. The point is that such candidate generalizations can be developed from scenario descriptions.

The generalization of the causal relations comprising scenarios can also be carried out across features. Speech annotation of data helps people create a personalized view of their information. But this relation holds independent of whether the data are annotated by speech, text, or handwriting. Understanding the relation more generally allows designers to consider any medium for annotation as a potential means of facilitating a personalized data view.

Scenarios can also be taken as exemplars of model scenarios; for example, box 3.1 illustrates a model of opportunistic control. Harry pursues the link from bridges to piccolos, because that is the aspect of the information that interests him. The system was designed to support this style of use; to that extent it embodies a model of opportunistic control. Other models are possible, of course; many instructional systems would require a student to complete the current module before allowing a branch to related material in some other module. Seeing the scenario in box 3.1 as an opportunistic control scenario allows the designer to benefit from prior knowledge pertaining to this model and to contribute further design knowledge of the model based on the current project.

Designers are not just making *things*; they are making sense. Particularly in technical design it is not enough for them to master a craft practice, for there is no stable craft practice in design domains like information systems and applications. The turbulence of technology development leaves little unchanged but, particularly in the short run, little resolved. We may expect that conventional science will eventually systematize technical knowledge in new domains, but we also know that it typically will do so after the wave of technological innovation has swept onward. The challenge for designers is to learn as they go.

## Toward a Scenario-Based Framework for Design

In the large, information technology appears as a coevolution of tasks and artifacts. People engage in tasks and activities. They make discoveries and encounter difficulties. They experience insight and satisfaction, frustration, and failure. At length, their tasks and experiences help to define requirements for future technology. Subsequent technological tools open up new possibilities for tasks and activities, new ways to do familiar things, and entirely new things to do. They entrain new skills to learn and perform, new opportunities for insight and satisfaction, and new pitfalls of frustration and failure. Ultimately the new tasks and new technology become a baseline, helping to evoke requirements for further technology development. This pattern, schematized in figure 3.1 as the task-artifact cycle, emphasizes the dynamically moving window of technology development within which technical design occurs (Carroll, Kellogg, and Rosson 1991).

The task-artifact cycle is pervasive in information technology. Consider the spreadsheet. The first electronic spreadsheet, VisiCalc, was brought to the market in 1979. It clearly embodied a simple yet powerful response to a set of extant tasks: table-based calculations. Placing the spreadsheet in an electronic medium permitted accurate calculation and convenient handling. And it did much more. It opened up important new possibilities for table-based calculation tasks. Electronic spreadsheets facilitated projective analyses (what-if reasoning about altered conditions and abductive reasoning from a desired result to conditions that could produce it). People using electronic spreadsheets could easily alter values and recalculate. Indeed, spreadsheets even afforded a kind of ad hoc work integration: people could type a memo describing an analysis right into a spreadsheet cell (Mack and Nielsen 1987).

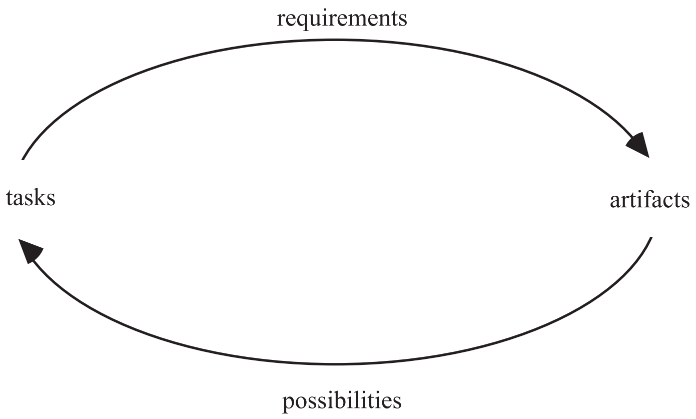

**Figure 3.1**
Task-artifact cycle. Tasks help articulate requirements for new technology that can support them; designed artifacts create possibilities (and limitations) that redefine tasks

This evolution of spreadsheet tasks can be viewed as successively altering requirements for spreadsheet systems. In the early 1980s, Context MBA provided integrated support for windows, graphing, word processing, and file management—for example, displaying the spreadsheet in one window and a graph of selected cells or a report in another. Lotus 1–2–3 introduced natural order recalculation (in which cell dependencies determine the order of recalculation), easing the overhead of what-if explorations. These advances, in turn, can be seen as encouraging further task evolution. For many office workers, the spreadsheet environment became a fulcrum for a myriad of activities: planning, communicating, accessing information, reporting, and presenting. It is now typical for spreadsheet systems to support multiple windows, integrate support for text and graphics, and share data with other programs. Many spreadsheets offer a range of recalculation options to facilitate projective analysis; some offer a "solver" function that takes a specification of a desired result and suggests how to obtain it.

Scenario-based design provides a framework for managing the flow of design activity and information in the task-artifact cycle. Scenarios evoke *task*-oriented reflection in design work; they make human activity the

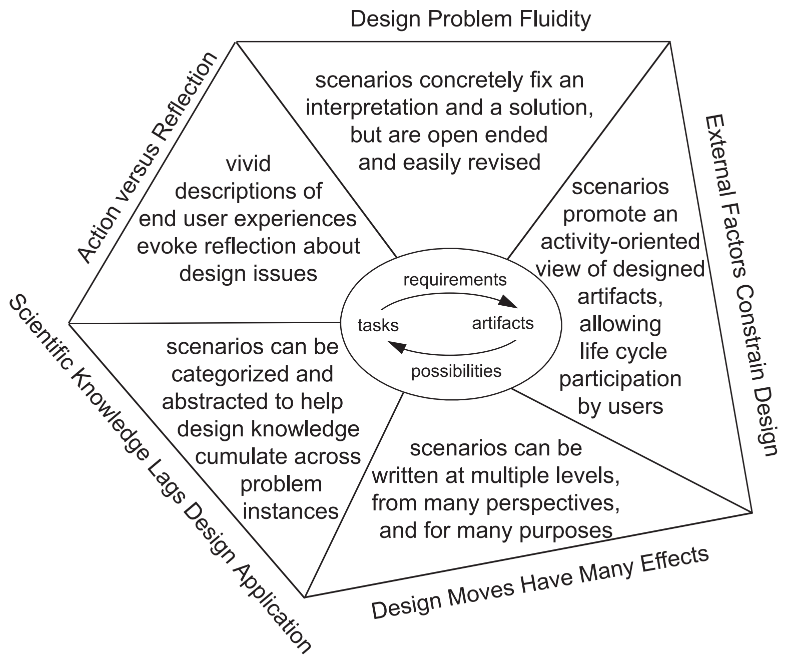
Design Problem Fluidity
External Factors Constrain Design
Design Moves Have Many Effects
Scientific Knowledge Lags Design Application
Action versus Reflection
scenarios concretely fix an interpretation and a solution, but are open ended and easily revised
scenarios promote an activity-oriented view of designed artifacts, allowing life cycle participation by users
scenarios can be written at multiple levels, from many perspectives, and for many purposes
scenarios can be categorized and abstracted to help design knowledge cumulate across problem instances
vivid descriptions of end user experiences evoke reflection about design issues
requirements
tasks
artifacts
possibilities

**Figure 3.2**
Challenges and approaches in scenario-based design

starting point and the standard for design work. Scenarios help designers identify and develop correct problem *requirements* by being at once concrete and flexible. They help designers to see their work as *artifacts-in-use* and, through this focus, to manage external constraints in the design process. Scenarios help designers analyze the varied *possibilities* afforded by their designs through many alternative views of usage situations. And scenarios help designers cumulate their knowledge and experience in a rubric of *task*-oriented abstractions. This is depicted in figure 3.2.

Design is a world that is part art and part science. It is a world of uncertainty and opportunity, a world in which sorcerers cast sweeping spells, often with great fanfare, though the spells generally fail. It is a world in which there is much talk of sound engineering practices, sometimes even of *formal* methods. Few, however, can bear to plod through the rote discipline of top-down structured decomposition toward what they know will be only mediocre results. Nevertheless, as managers and teachers, they may even encourage others to follow such methods, knowing that they will not do so, that they probably cannot do so.

To manage messes, we must come down from the clouds of pedagogical engineering and enter the swamp of real activity—in which new domain knowledge must be discovered in order to understand needs and objectives, in which design moves must be concrete to be evaluated and revised, in which every move taken potentially affects every other move, and in which diverse skills and perspectives are required to plan, execute, and evaluate every move. It is an interesting swamp; most designers would not wish to live anywhere else. We may come to know it better and cope with it better through a more explicit scenario-based design practice.