Five Reasons for Scenario-Based Design

John M. Carroll
Department of Computer Science and
Center for Human-Computer Interaction
Virginia Tech
Blacksburg, VA 24061-0106
Tel: 1-540-231-8453
E-mail: carroll@cs.vt.edu

Abstract

Scenarios of human-computer interaction help us to understand and to create computer systems and applications as artifacts of human activity —as things to learn from, as tools to use in one's work, as media for interacting with other people. Scenario-based design of information technology addresses five technical challenges: Scenarios evoke reflection in the content of design work, helping developers coordinate design action and reflection. Scenarios are at once concrete and flexible, helping developers manage the fluidy of design situations. Scenarios afford multiple views of an interaction, diverse kinds and amounts of detailing, helping developers manage the many consequences entailed by any given design move. Scenarios can also be abstracted and categorized, helping designers to recognize, capture, and reuse generalizations, and to address the challenge that technical knowledge often lags the needs of technical design. Finally, scenarios promote work-oriented communication among stakeholders, helping to make design activities more accessible to the great variety of expertise that can contribute to design, and addressing the challenge that external constraints designers and clients often distract attention from the needs and concerns of the people who will use the technology.

1. Introduction

Designers of information systems and applications face a disturbing reality. While there is plenty of opportunity to do things that make a difference, it is never unequivocal just what should be done, or even just what the real problems are. The problems can only be definitively analyzed by being solved; the appropriate solution methods must typically be executed in order to be identified; the solutions must be implemented in order to be specified. All the while, the designer faces convoluted networks of tradeoff and interdependency, the potential of untoward impacts on people and their social institutions, and the likelihood that changing cultural and technological circumstances will obviate any solution before it can be deployed.

Most software engineering methods belong to a 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. A complementary tradition seeks to *exploit* the complexity and fluidity of design by trying to learn more about the structure and dynamics of the problem domain, by trying to see the situation in many different ways, and by interacting intimately with the concrete elements of the situation [1,2,11,28].

Scenario-based design techniques belong to this complementary approach. 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 tradeoffs to reach usable and effective design outcomes [5,6].

2. 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 computer technologies. In analyzing and designing systems and software we need better means to talk about how they may transform and/or be constrained by the contexts of user activity: this is the only way we can hope to attain control over the "materials" of design. A direct approach is to explicitly envision and document typical

0-7695-0001-3/99 $10.00 (c) 1999 IEEE

and significant user 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. They are 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 even this scenario 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 [26]. They include or presuppose a *setting*: The accountant scenario explicitly describes a starting state for the described episode; the relative positions of the folder and spreadsheet, and the presence of the accountant. The scenario implies further setting elements by identifying the person as an accountant, and the work objects as budgets and memos.

Scenarios also 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 agent or actor typically has *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 accountants 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 user 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 user-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.

In the balance of this chapter, we review five of key challenges for design methods and illustrate for each the corresponding response of scenario-based design.

3. Challenge: Design action competes with reflection

Technical professionals are intelligent people 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. There is a fundamental tension between thinking and doing: thinking impedes progress in doing, and doing obstructs thinking. Sometimes this conflict is quite sharp, as when one must stop and think before taking another step. But frequently it is more a matter of trading off priorities.

Donald Schön [28,29] has discussed this conflict extensively in his books on reflective practice. For example, he analyzes a coach reacting to an architecture student's design concept for a school building; the design includes 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." [29, page 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. However, he does not bother to justify this advice to the student, to allow her to see the rationale behind his advice; as Schön puts it, he does not reveal "the meanings underlying his questions" [29, page 132]. Schön regards this as a hopeless confrontation in which no progress can be made on the particular design project, or on the larger project of understanding how to design. Both the student and the coach are willing to act publicly and to share actions, but do not reflect enough on their own and one another's values and objectives, and on their interpersonal dynamics.

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 [29] 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)

In general, people do not like to make themselves self-aware of their own roles as actors in situations; indeed, being "objectively self-aware" impairs performance of nonroutine tasks [14] — like designing software!

4. Scenarios evoke reflection in design

Designers do try to create opportunities for their own reflection. They organize design review meetings in which the whole team works through a set of requirements, a progress report, or a specification. It is also common to build early prototypes to verify and refine design requirements; one can directly observe prospective users interacting with such a prototype to make a formative evaluation [30] of the design. These activities can facilitate the identification and integration of different perspectives; they can raise concrete and detailed design issues to guide further work. In this way they help designers to reflect on the work they've already done. However, they do not evoke reflection *in the context of doing design*. Though design reviews and formative evaluations are reflective activities, they are ancillary activities that must be coordinated with design itself. Prototyping is directed at building and testing software that embodies a design, not on thinking about the design as it is produced.

Constructing scenarios of use inescapably evokes reflection in the context of design. Consider the scenario in Figure 1: it 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 or a list of requirements. Elements of the envisioned system appear in the scenario embedded in the user interactions that will make them meaningful to the user — 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.

> Harry is interested 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 opens 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.

Figure 1: A usage scenario for a multimedia education project

The scenario emphasizes and explores goals that the user may 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 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 user's actions and experiences.

The scenario concretely embodies a partial view of the design, and thereby exposes the design to critique. In this sense, the scenario as an object can function much like a "soft" prototype (though, as we have suggested, the *process* of designing a scenario more strongly evokes reflection). Indeed, scenarios have been found to be extremely useful as focal objects in both design reviews and formative evaluations [7,13,21]. The scenario exposes not only the functionality of the system, but specific claims about how the user will access that functionality and what the user will experience in doing so.

Schön [28, page 67-68] drew an important contrast between merely creating or identifying elements of the problem context, and "giving them reason." Practitioners, as experts, will often have a category or a label with

which to "understand" a situation. In reality, this simple step often establishes an insurmountable obstacle to achieving a real understanding. "When someone reflects-in-action, he becomes a researcher in a practice context." (p. 68) A state of subjective self-awareness is encouraged by designing scenarios because the focus of attention is the activities and experiences of the prospective user.

5. Challenge: Design situations are fluid

Design analysis is always indeterminate, because design changes the world within which people act and experience. The rapid evolution of spreadsheet software in the 1980s does not indicate a failure in the original requirements analysis for VisiCalc, but rather suggests the extent to which the original spreadsheet programs altered the work situations in which these program were used [9]. Requirements always change [3]. When designs incorporate rapidly-evolving technologies, requirements change even more rapidly. The more successful, the more widely-adopted and the more impactful a design is, the less possible it will be to determine its correct design requirements. And in any case, refinements in software technology and new perceived opportunities and requirements propel a new generation of designs every 2-3 years.

There is a tendency to think of the indeterminacies of technology development in positive terms: the world is getting better, albeit sometimes in ways we didn't expect. Recently this positive attitude has been supplemented with growing acknowledgment of potential negative consequences, such as pollution of groundwater and the deterioration of the atmosphere. But both these emphases underanalyze the extent to which design, and especially the design of new technology, *undermines the stability of the world*. Schön [27], for example, noted that in our era technology development constantly erodes the 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. Even writers, such as McLuhan [23] who revel in the prospect of "learning a living," agree about the magnitude of this instability.

The instability of design situations originates inside design teams as well as outside. As a project goes forward, its funding may be threatened or restructured; various stakeholders or team members may change their interests or priorities, or may even leave the team. Others with unknown interests and priorities may join. This creates a chronic need for education and consensus-building to ensure that there is agreement as to what the requirements are at any given point in time.

Schön [27] stressed that in the face of great instability, people create illusions of stability to manage their own uncertainties and potentially disturbing perceptions. This is consistent with a substantial body of social psychology research: People create logically tidy interpretations of their experience, even if they must "adjust" their actual perceptions in order to do this [25]. People protect their interpretations by selectively reconstructing their own experience [15]. And the more reconstructionist effort people must expend to maintain an interpretation, the more ardently it is held, even in the face of incontestable disconfirmation [16].

Given the human bias for stability, we should not be surprised by the persistence of the attitude that design problems can be planfully decomposed into routine subproblems, that scientific principles can be mechanically applied to expose and manage the underlying orderliness of design problems. From the standpoint of the psychology of designers and managers, it could conceivably become even stronger as it becomes more clearly inappropriate [17]. These dysfunctional beliefs about design would not be such a problem if they could only lead to standstill. The potentially dangerous aspect is that they can lead to "success" in accomplishing the wrong things. Designers almost always design something. If their need for stability induces them to prematurely close their designs, to cease reflecting and critiquing, to declare victory, they may produce a solution for requirements that no longer exist.

6. Scenarios are at once concrete and flexible

To manage an ambiguous and dynamic situation, one must be concrete but flexible. One must be concrete just to avoid being swallowed by the indeterminacies; one must be flexible to avoid being captured by a false step. Systematic decomposition is a traditional approach to managing ambiguity, but it does not afford flexibility. Instead one ends up with a set of concrete subsolutions, each of which is fully specified. Unfortunately, by the time the set of subsolutions is specified, the requirements often have changed.

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 Figure 1 specifies a particular usage experience that could be prototyped and tested. At the same time the scenario is flexible, deliberately incomplete and easily revised or elaborated: in a few minutes, a piece of the scenario could be re-written (e.g., perhaps the associated module opens automatically) or elaborated (e.g., the module may be opened by following a "related materials" tag attached to the film clip).

A scenario provides a concrete envisionment of a design solution, but can be couched at many levels of detail. Initial scenarios are typically very rough. They specify a system design by specifying the tasks users can or must carry out, but without committing to the lower-level details of precisely *how* the tasks will be carried out or *how* the system will enable the functionality for those tasks. The narrative in Figure 1 is at an intermediate

level, with some detail regarding task flow, but not at the level of individual user-system interactions.

Thus scenarios provide a stable foundation for action-oriented reflection. By being both concrete and rough, they make explicit the design goal of specifying tasks and functions in greater detail. But they do this by opening up the issue of how to do this detail design, instead of closing it out prematurely. Scenarios allow designers to provisionally construct a space of user tasks despite the instability in requirements originating from the context of technology development. Scenarios are particularly appropriate for managing the instability originating inside design teams: They are broadly accessible to various stakeholders and team members, unlike artifact-oriented decompositions such as functional specifications.

The notion that scenarios of use are an appropriately concrete but rough design object extends several earlier analyses. For example, Schön [28, page 277-279] emphasized that effective reflection must be tightly coupled to action: the analysis need not be complete and consistent, it need only guide a restructuring of the current situation that can produce new design actions or new insights. His cases are drawn from design domains for which there are rich languages to allow designers to quickly create and explore situations. Scenarios are such a language for the design of human-computer interactions.

In an earlier book, Schön [27, page 41] argued that decomposition is useful chiefly through the side-effects of eliciting and focusing concrete action, and thereby reducing uncertainty. But he warned that the resultant decomposition can "strangle" innovation. Ackoff [1] similarly developed the notion of "idealized design": a process of planning "the system with which the designers would replace the existing system *now* if they were free to do so."(p. 191) He argued that such a process can increase participation among stakeholders, mobilize participants with respect to the project, expand the concept of what is feasible, increase both creativity and the scope of human and organizational consequences that are considered in the design process; an idealized design is really just a vehicle for focusing the articulation of possibilities and the discussion of their consequences.

7. Challenge: Design moves have many consequences

Every element of a design, every move that a designer makes, has a variety of potential consequences. In their work on violin design, the Catgut Society found that additional rib holes were needed in their new treble violin to pitch the instrument high enough and yet maintain the neck length for fingering. This design move had the intended consequences, but also forced the designers to consider stronger materials for the ribs, and ultimately to use *aluminum* instead of wood. However the aluminum ribs caused a nasal quality in the lower strings, forcing the designers to reconsider a new wooden rib design — ribs so thin (0.7 mm) that the designers believed on analytical grounds that they would collapse. A typical complication in this design case is that the various consequences belonged to different stakeholders: A fingerboard that is too short is a problem for the musician. A very thin wooden rib is a problem for the instrument maker. A nasal tone is a problem for the musician and listener [19].

Schön [28, page 101] sees design as a "conversation" with a situation comprised of 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 Catgut group made the move of employing aluminum ribs and then noticed the unanticipated problem of a nasal tone in the lower strings. 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" [28, page 155].

Schön's approach to managing the interdependencies within a design situation by treating design as inquiry raises several questions. What directs the designer to investigate various types of consequences, to listen for various types of backtalk? How are different types and sources of backtalk integrated in the designer's restructured design theory? The moves for the treble violin had various consequences. The Catgut group effectively integrated these in their subsequent design work, but how did they do that and, more generally, how can we support outcomes like that? Designers need a language for this conversation and they need techniques for managing the consequences.

8. Any scenario has many views

Scenarios of use are multifarious design objects; they can describe designs at multiple levels of detail and with respect to multiple perspectives. The scenario in Figure 1 provides a high-level task view, but it easily could be elaborated with respect to the detailed moment-to-moment thoughts and experiences of the user in order to provide a more detailed cognitive view, or the user's moment-to-moment actions to provide a more detailed functional view. Alternatively, it could be elaborated in terms of hardware and software components that could implement the envisioned functionality in order to provide a system view (use cases play this role in object-oriented software engineering [20,32]. 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. In Figure 1, the envisioned speech annotation capability allows the user to add a personal comment without the overhead of opening an editor and typing text. However, the annotation is noncoded, and thus cannot be edited symbolically. These tradeoffs are important to the scenario, but often it is enough to imply them (this is an aspect of the roughness property discussed above).

There are times, however, when it is useful to make these relationships explicit. For example, in another situation 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, address, and track 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 the scenario. Doing so provides yet another view of the envisioned situation as shown in Figure 2.

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
- but may not sufficiently motivate a learners to attempt the course module

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

Figure 2: A view of the multimedia education scenario sets of consequences associated with the orientational video clip and the speech annotation capability.

Scenarios can help designers move toward consequences of differing types. For example, the data structures for the user's workbook might differentiate between annotated and non-annotated items, allowing annotated items to be retrieved and browsed as a subset. This would not allow Harry to directly retrieve the set of items with a particular annotation, but 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 Figure 1 and the search scenario discussed above. 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% recognition accuracy. One would want to investigate tradeoffs 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 user would be aware when creating the annotations that they will later be retrieval keys. Providing a finely articulated data structure for the user's 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 to 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.

9. Challenge: Technical knowledge lags technical design

Schön [28, page 42-44] 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 only address 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 technical knowledge systematically, to reduce difficult problems to mere corollaries — and of course to bask in the confidence that deductive science confers both on its practitioners and the recipients of their efforts: science is certainty. But at the same time, technology design and development is inevitably driven to pursue novelty and innovation, and thereby to slip continuingly into the swampy unknown regions.

Schön [28, page 16, 42-44] discusses the field of operations research as an example. 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, or criminal justice. Remarkably, the response of the field to this was largely to focus attention on the relatively simple problems that suited the mathematical models.

Russell Ackoff, one of the founders of operations research, lamented the inability and disinterest of the field in addressing the "turbulent environments" of the real world [2, page 94]:

> "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." [2, page 99-100]

This quote 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." [2, page 100].

Our own experience in instructional design illustrates this pattern very well. Modern instructional design blossomed in the 1960s and subsequently in response to rapidly expanding needs for technical training [31]. The basic model that emerged and guided this work was based on hierarchical task analysis: the overall instructional objectives were successively decomposed into enabling objectives, objectives that enabled the enabling objectives, and so on; at the lowest level, these subskills were described, drilled and tested [18]. The vision of this "systems approach" model is that the instructional designer carefully orchestrates a series of well-controlled instructional events that build up the learning hierarchy within the student's mind.

At the time this model was widely adopted, three decades of research on learning and education had already made it clear that no one ever learns this way. People can learn in spite of such an approach, but only because they are so adaptable. The whole enterprise seems to have been configured to ignore the most difficult issues of learning and education and to reliably produce mediocre results. In the 1970s and early 1980s, this approach was pervasive in the design of instruction for computer systems and applications, and was found to be particularly ineffective for non-programmers. Unfortunately, non-programmers were the most rapidly growing segment of computer users during those years. It turned out to be possible to design effective instruction for these users, by studying and supporting actual situations of learning, instead of merely asserting the problem and the solution in a vacuum of systematic decomposition [4].

The critical elements for an effective learning experience are activity and discovery that can be recognized as meaningful *by the learner* and as supporting current personal interests, needs, and objectives *of the learner*. People want to learn in a realistic context; they want to be able to use what they already know to critically evaluate new knowledge and skills they encounter. These themes are not new discoveries, they had been widely developed by psychologists like Jerome Bruner, John Dewey and Jean Piaget and were well-known in 1970. But they do not admit of simple assembly-line instructional designs. Taking them seriously makes instructional design a very open-ended process; it guarantees little about either the types of design analysis that will be required in a given case or the types of instructional designs that will emerge as appropriate and effective in a given case.

10. Scenarios can be abstracted and categorized

Though technical design cannot escape Schön 's swamp, designers need to extract lessons from their experience to guide their work and improve their practice. If technical knowledge lags design, then designers themselves must formulate what they learn — perhaps becoming creators of technical knowledge more than consumers of it. The Catgut Society is a beacon for this: At the start of their project there was little relevant technical knowledge; physicists had a good understanding of vibrating strings, but no significant understanding of vibrating systems composed of strings vibrating across air cavities partitioned by wooden ribs and bounded by irregularly-shaped wooden boxes. The Catgut work in the end has substantially advanced technical knowledge of complex acoustical systems: the design work created an "island" of hard ground in the swamp of real problems.

Scenarios keep the designer of computer systems and applications in the swamp, but by their very nature also provide scaffolding to get a view of the design situation from a bit higher up. The roughness of scenarios is, after all, abstractness: To the extent that a scenario is rough, it is a description of a space of particular user interactions. This is analogous to the way in which a jazz score is a rough description of the many ways in which that piece of music might be performed. 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 in Figure 1 could guide the design of a multitude of information systems.

But this is only the most primitive technique for developing design knowledge with scenarios. Scenarios exemplify particular themes and concerns in work and activity situations. Earlier we discussed two scenario for a multimedia education system. In one (Figure 1), the user is at first opportunistically exploring an information structure, but eventually adopts a particular interest that guides his exploration. In the other, the user 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 in the subsequent design of other situations. Such a taxonomy of scenarios provides a framework for developing technical design knowledge.

Scenarios can also be classified in terms of the causal relations they comprise. In Figure 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 only be settled 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 the user create a personalized view. But this relation holds independent of whether the data is annotated by speech, by text or by 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, Figure 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 Figure 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 computer systems and applications. The turbulence of technology development leaves little unchanged, but particularly in the short run, leaves 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 onwards. The challenge for designers is to learn as they go.

11. 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, are clearly the best source of constraints because they indicate what sort of design work is needed. But there are many other sources of constraints. The current state of technology development makes some solutions impossible and others irresistible: On the one hand, designers cannot use technology that does not yet exist, though their work often drives technology development toward possibilities that are nearly within reach. On the other hand, designers, like everyone else, are caught up in a technological zeitgeist that biases them toward making use of the latest gadgets and gizmos. In addition, designers are often biased toward deploying technologies they have used before, even when they are aware of limitations in these technologies.

Earlier we referred to our work in instructional design. In this domain, we found that many designers continued to use the systems approach even though they knew that it was inappropriate for their users. They did this because the approach ensured a uniformly structured, comprehensive result that other instructional designers could recognize as well-executed. This aligned the designer with others employing the same model, creating a professional community. The many difficulties the approach created for learners were viewed as merely part of the research agenda, although in fact few of these problems received adequate attention.

As Churchman [12] observes, many systems are designed to serve the "wrong" client, for example, hospitals are designed to serve doctors, not patients. In time, of course, a system designed to serve the wrong client will poorly serve all clients. This is similar to the confusion of "customers" and "users" in identifying design requirements. However, by far the worst such confusion is entrained by solving tidy problems instead of addressing the real problem. In the systems approach to instructional design, the mistaken client is the instructional designer, or perhaps the instructional design firm; the underlying intent is to streamline the production of instruction, not to improve its utility.

Many constraints in design originate in the organizational structures within which design work is embedded. Only certain types of assumptions and arguments are acceptable in the business cases that underwrite design projects. For example, well into the 1980s it was widely believed that executives would never use a keyboard. This belief was based on a marketing stereotype of executive disdain for clerical tasks, yet it constrained many technical strategies for the development

of computer hardware and software in that period. Ultimately, appropriate hardware and software were designed for executives, as one can easily verify in the number of notebook computers in use in the first class cabin of an airplane. Apparently, the issue had nothing to do with typing per se, but with other aspects of use situations.

Power structures and stereotypes play assorted other roles. Design projects are often chartered with a priori commitments to follow a strict decompositional approach: which makes them easy to manage, but unlikely to succeed with respect to serving the needs of users and clients. In such a context, designers must struggle not only with technical challenges but with an impossible methodology. Schedules and resources are often assigned in ways that create on-going conflicts between system designers and usability engineers: the usability engineers need to evaluate scenarios and prototypes at every stage of system development, but if schedules and resources do not provide for this, the usability work can conflict with development work.

In all these examples of external constraints, the designers can become distracted by ancillary or even perverse factors and lose sight of what is essential in the design project, namely, the needs and concerns of users. The designer can become "unsituated" with respect to the real design situation, which is not the marketing manager's projections, or the instructional designer's list of steps, or the software engineer's system decomposition. The real design situation is the situation that will be experienced by the user, and designers need to stay focused on that.

12. Scenarios promote work-orientation

Scenarios are work-oriented design objects. They describe systems in terms of the work that users will try to do when they use those systems. A design process in which scenarios are employed as a central representation will ipso facto remain focused on the needs and concerns of users [8].

One can increase the effectiveness of scenarios of use as work-oriented design objects by couching them at an appropriate level and by directly involving users in creating them. A person's experience of working with a system revolves around understanding and developing task goals, responding opportunistically to intriguing system events, and planning, carrying out and evaluating courses of action. People do not focus on the very low-level enabling actions and events that comprise these experiences, such as keypresses, beeps, mouse gestures and clicks, display updates, and font size. Quite often people are only aware that they are seeing, hearing and doing these things when something goes amiss. A heuristic for writing scenarios focused on the client's needs and concerns is to initially couch them at the "basic task" level — the level at which people experience their own activity [10].

Ackoff [1,2] argued that the indeterminacy of design situations made it imperative that *all* stakeholders participate directly. Assuming that stakeholders can represent their own interests, this proposal pretty much guarantees an adequate focus on client's needs and concerns. However, it may not be the case that all stakeholders can represent their interests: they will be trying to evaluate their needs as transformed by new technologies that they may not understand. Moreover, it is not always feasible to include all stakeholders; in the design of a new spreadsheet application for personal computers there might be several million stakeholders.

Ackoff's ideas have been refined through various developments in participatory design over the past two decades. One technique is to include representative clients on the design team, instead of imagining that every client can participate directly. Of course client representatives on participatory design teams are not representative clients; by definition only clients not on the design team can be truly representative. However, given this dilemma, "representative" sampling may be a reasonable heuristic. Clients can also be helped to better represent their interests in design collaborations by taking part in facilitated demonstrations of possible scenarios of use [22,24].

13. Some final words

Our objective in this paper was to motivate and preview a framework for managing design that accommodates the nature of design problem solving as it occurs in the context of technology development. Our approach tries to facilitate flexible design actions informed by reflection on multiple levels and from multiple perspectives, including direct collaboration among team members. We argue that making scenarios of use a focal design object serves this variety of purposes.

Figure 3 summarizes the five issues and corresponding approaches we discussed. The key issue is encouraging, supporting, and productively directing reflection that is closely and effectively integrated with action in the design process. Designers want to reflect, but they know from experience that it is impossible to fathom all the consequences and interdependencies. At the same time, they know that they can act and immediately see progress toward a solution, or at the least feel that they have eliminated some possibilities. Faced with this choice, it is always more attractive in the short-term to act. In our work on instructional design, we noticed an analogous conflict in the activities of learners that we called the "production paradox" [4]: People know that they must learn new concepts and skills in order to be able to do new sorts of things, however, they also know that by just trying things out they can see and feel progress, learning as they accomplish something meaningful.

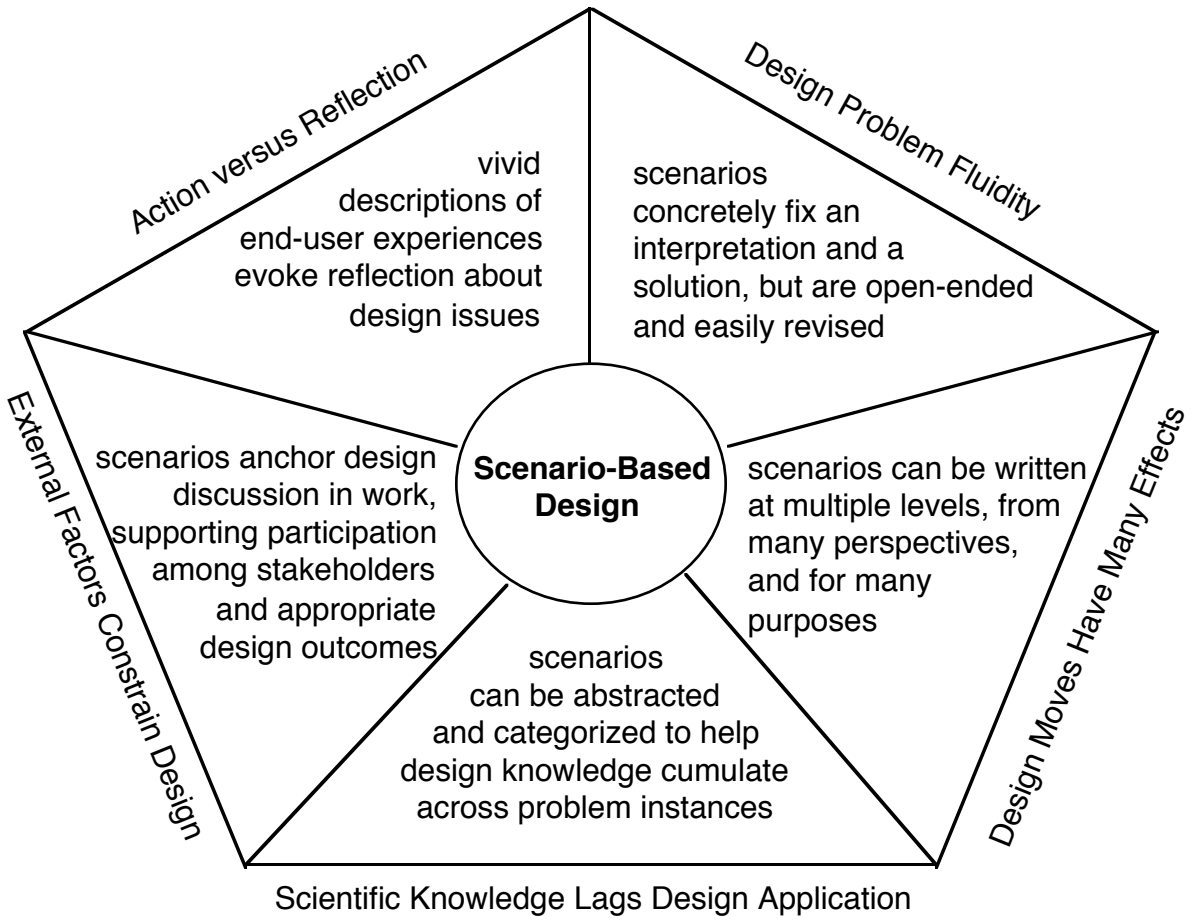

Figure 3: Challenges and approaches in scenario-based design

Scenarios of use help designers manage the production paradox. Creating and elaborating scenarios is concrete design work; the designer sees and feels progress toward a design result. At the same time, scenarios are concrete hypotheses about what the people using the design result will do, think and experience. Thus, in Schön 's [28] terminology, scenarios evoke reflection-in-action. Schön stressed the importance for designers to experience the "felt-path" of the people interacting with their designs. A scenario guarantees this experience: it presents a potential felt-path to the designer; it is a medium through which designers can envision and explore alternative felt-paths.

Scenarios evoke effective reflection in a way that addresses some of the most difficult properties of design. The fluidity of design situations demands that solutions be provisional, that commitments be tentative; yet if every design decision is suspended, the result will be a design space, not a design. A scenario is a concrete design proposal that a designer can evaluate and develop, but is also rough in that it can be easily altered and allows many details to be deferred.

The interconnectedness of design decisions, and the variety and extent of any given decision's consequences, requires designers to consider their decisions from many different perspectives: software architecture, marketing, ease of learning, production cost, usability, and so forth. It requires them to consider their decisions at many different levels of detail in the proposed solutions. Scenarios of use serve as a concrete context for developing and integrating these different perspectives and levels.

Technical innovation is driven to those regions where technical knowledge is thin. Designers often have little more than craft practices to guide them in these regions. But if we see design as inherently a process of inquiry, this lag between codified knowledge and practice is transformed into a significant opportunity. Design can be a paradigm for creating and cumulating technical knowledge. Scenarios provide a broad rubric for organizing and generalizing knowledge attained in design contexts. They can be abstracted and categorized in various ways, and then employed in new design problems.

Like every human activity, design work occurs in many overlapping social contexts: technical societies, corporations, states of technology development, industries, and so forth. Each context introduces constraints on possible methods and solutions.

Sometimes these are constructive and appropriate, often they are neither. The cacophony of these social contexts and constraints can leave designers unsituated with respect to their primary source of constraint — the needs and concerns of the people who will use the system being designed. Scenarios keep the whole enterprise focused on past and future situations of use, they help keep designers focused on what matters most.

References

[1] Ackoff, R.L. "Resurrecting the future of operations research" Journal of the Operations Research Society, 30(3), 1979, pp. 189-199.

[2] Ackoff, R.L. "The future of operations research is past", *Journal of the Operations Research Society, 30(2),* 1979, pp. 93-104.

[3] Brooks, F. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, Reading, MA, Anniversary Edition 1995 (originally 1975).

[4] Carroll, J. M. *The Nurnberg Funnel: Designing Minimalist instruction for practical computer skill*. MIT Press, Cambridge, MA, 1990.

[5] Carroll, J.M. "Making *use* a design representation", *Communications of the ACM, 37*/12, 1994, pp. 29-35.

[6] Carroll, J.M., Ed. *Scenario-based design: Envisioning work and technology in system development*. John Wiley and Sons, New York, 1995.

[7] Carroll, J.M. & Rosson, M.B. "Usability specifications as a tool in iterative development", in H.R. Hartson (Ed.) *Advances in Human-Computer Interaction*. Ablex, Norwood, NJ, 1985.

[8] Carroll, J.M. & Rosson, M.B. "Human-computer interaction scenarios as a design representation", in *Proceedings of the 23rd Annual Hawaii International Conference on Systems Sciences*. (Kailua-Kona, HI, January 2-5, 1990). Los Alamitos, CA: IEEE Computer society Press, 1990, pages 555-561.

[9] Carroll, J.M. & Rosson, M.B. "Deliberated evolution: Stalking the View Matcher in design space", *Human-Computer Interaction, 6,* 1991, pp. 281-318.

[10] Carroll, J.M. & Rosson, M.B. "Getting around the task-artifact cycle: How to make claims and design by scenario", *ACM Transactions on Information Systems, 10,* 1992, pp. 181-212.

[11] Checkland, P.B. *Systems thinking, systems practice*. Wiley, New York, 1981.

[12] Churchman, W. 1970. Operations research as a profession. *Management Science, 17(2),* 37-53.

[13] Chin, G., Rosson, M.B. & Carroll, J.M. "Participatory analysis: Shared development of requirements from scenarios", in S. Pemberton (Ed.), *Proceedings of CHI'97: Human Factors in Computing Systems*. (Atlanta, 22-27 March). ACM Press/Addison-Wesley, New York, 1997, pp. 162-169.

[14] Duval, S. & Wicklund, R.A. *A theory of objective self-awareness*. Academic Press, New York, 1972.

[15] Erikson, E.H. *Identity and the life cycle*. Norton, New York, 1980.

[16] Festinger, L. *A theory of cognitive dissonance*. Harper & Row, New York, 1957.

[17] Festinger, L., Riecken, H.W. & Schachter, S. *When prophecy fails*. University of Minnesota Press, Minneapolis, 1956.

[18] Gagne, R.M., & Briggs, L.J. *Principles of instructional design*. Holt, Rinehart and Winston, New York, 1979.

[19] Hutchins, C.M. "The acoustics of violin plates", *Scientific American, 245(4),* 1981, pp. 170-186.

[20] Jacobson, I. "The use-case construct in object-oriented software engineering", in J.M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development*. John Wiley & Sons, New York, 1995, pp. 309-336.

[21] Karat, J. & Bennett, J.B. "Using scenarios in design meetings – A case study example", in J. Karat (Ed.), *Taking design seriously: Practical techniques for human-computer interaction design*. Academic Press, Boston, 1991, pp. 63-94.

[22] Kyng, M. "Creating contexts for design", in J.M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development*. John Wiley & Sons, New York, 1995, pp. 85-107.

[23] McLuhan, M. *Understanding media: The extensions of man*. MIT Press, Cambridge, MA, 1994 (original edition, 1964).

[24] Muller, M.J., Tudor, L.G., Wildman, D.M., White, E.A., Root, R.A., Dayton, T., Carr, R., Diekmann, B., & Dystra-Erickson, E. "Bifocal tools for scenarios and representations in participatory activities with users", in J.M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development*. John Wiley, New York, 1995, pp. 135-163.

[25] Nisbett, R.E. & Wilson, T.D. "Telling more than we can know: Verbal reports on mental processes", *Psychological Review, 84*, 1997, pp. 231-259.

[26] Potts, C. "Using schematic scenarios to understand user needs", in *DIS'95: ACM Symposium on Designing Interactive Systems*, (Ann Arbor, MI). ACM Press, New York, 1995, pp. 247-256

[27] Schön, D.A. *Technology and change: The new Heraclitus*. Pergamon Press, New York, 1967.

[28] Schön, D.A. *The reflective practitioner: How professionals think in action*. Basic Books, New York, 1983.

[29] Schön, D.A. *Educating the reflective practitioner*. Jossey-Bass San Francisco, 1987.

[30] Scriven, M. "The methodology of evaluation", in R. Tyler, R. Gagne, & M. Scriven (Eds.), *Perspectives of curriculum evaluation*. Rand McNally, Chicago, 1967, pp. 39-83.

[31] Schriver, K. *Dynamics in document design*. John Wiley and Sons, New York, 1997.

[32] Wirfs-Brock, R. "Designing objects and their interactions: A brief look at responsibility-driven design", in J.M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development*. John Wiley & Sons, New York, 1995, pp. 337-360.