Rhetorics and Technologies

Stuart A. Selber, Carolyn A. Miller

Published by University of South Carolina Press

Narrating the Future

Scenarios and the Cult of Specification

John M. Carroll

It is a touchstone of contemporary culture that we invent the future. I was originally trained in the canon of early cognitive psychology and generative linguistics. But I have lived my career among people who do this literally—computer scientists, software engineers, and information technologists.

The activity of inventing the future is centrally about anticipating needs and interests of human beings. Not surprisingly, just what those future needs and interests will be is always unclear. When I was a graduate student, Alan Kay was inventing the Dynabook, the concept of a notebook form factor for personal computing. At the time, most people were still daunted by desktop terminals directly wired to gargantuan mainframes. Indeed, for years people scratched their heads about Alan Kay, but now it is clear that the only thing he really had wrong was the decade. He invented the future; it is our present.

In information technology, the Dynabook may stand out, but it is just a very sharp example of something that happens all the time. A more recent case is the MP3 player, most obviously, of course, the iPod. In a few short years, this device has been transformed from a slightly exotic item marking its owner as a cool, though most likely decadent, college student into a standard artifact of contemporary culture.

Technology is not only about increasing joy by fulfilling possibilities. It is also and often about perpetrating and then mitigating agonies. It is about managing problems. Technology development can be seen as iterating cycles of problem solving and problem spawning. At the time Alan Kay was dreaming about the Dynabook, many of his contemporaries were staring at blinking cursors, trying to guess what arcane command might coax the computer into making an understandable response. This was the so-called recall problem of early

command line user interfaces. People are not especially good at recalling arbitrary command names, names of files, and complex labels. And recall difficulties are heightened when the recall target is downright cryptic, like the critical Unix command *grep* (for "search Globally for lines matching the Regular Expression, and Print them"; Carroll, *What's in a Name*).

Every problem in technology has many solutions. For example, the recall problem (among others) entrained a new sort of user interface that presented labeled document icons scattered about on a metaphorical "desktop." One accessed and manipulated the properties and the functions of these objects via menus. These innovations directly address the recall problem: People no longer had to recall file names or command names. They could rely on the graphical user interface to present these directly. They could rely on *recognition* memory instead of *recall* memory. People are extremely good at recognition.

Every solution eventually entrains its own boundary conditions. The graphical user interface seems like a great idea when one cannot recall a file name but can see the file plainly on the desktop. But after a while, when there are several hundred or several thousand files on the desktop, the idea is not quite so brilliant. In this way, each new solution entails a new set of problems, and further design and invention. I call this canonical pattern the "task-artifact cycle" (Carroll, *Making Use*).

In this essay I discuss scenarios—brief and evocative narrative descriptions—as a design representation. The bright side of my argument is that by describing and analyzing information technology designs through narratives of their use (that is, *before* those designs are ever implemented and deployed to users) future problems and possibilities can be anticipated and managed. The dark side of my argument is that traditional, specification-based design methods minimize the chance of anticipating problems or of achieving possibilities. From this, I conclude that information technology design should be construed and developed as a rhetorical practice and not merely as a systems and software engineering practice. We need to cultivate methods for narrating the future.

Specifications

How can we get from imaginable possibilities and currently experienced problems to the future we want to invent? There is—always—an established approach, the establishment as it were. In the engineering disciplines, including computer science, software engineering, and information technology, the established approach is specification. A specification is a structured analysis of the parts and relationships that comprise a complex object.

A typical use of specification is functional specification, in which one enumerates the components and properties of a piece of functionality, for example, a command or a set of related commands in a software system. The functional

specification defines a software component in terms of its parts and properties, its relationships to other bits of functionality, and information about how to operate it and how it is implemented. Clearly specifications are a useful sort of design representation, and they are critical documentation if one ever needs to repair, extend, or refactor an existing piece of software (as one nearly always does). Specification is used pervasively in the design of hardware, software, and even human activity systems, where it is called task analysis.

An example of a specification. It describes the Smalltalk inspector tool (Carroll and Rosson). The basic functions, the component parts and their properties, and the interactions with other Smalltalk capabilities are listed. A more detailed version of this specification would probably include a screen shot, because the inspector is a tool for a graphical user interface system.

Functional Specification for a Smalltalk/V Inspector

The Inspector is a low-level debugging aid used to examine and edit objects.

Inspector components:

The *instance variable list* and the *instance variable contents.*

The *instance variable list* appears in a list pane positioned as the left pane of the tool. The first instance variable in the list is self, the object being inspected. The *instance variable contents* appear in a text pane on the right of the tool. This pane displays the contents of the variable currently selected in the list pane.

There is a special Inspector for Dictionary objects: the instance variable list contains Dictionary keys, rather than named or indexed instance variables. The special variable self is not included in the Dictionary Inspector variable list. There is also a special "method context" Inspector used by the system debugger (see below).

Inspector properties:

Browsing the instance variable list: Instance variables (including self) can be selected (clicked on) in the list pane; this causes the selected name to highlight and the contents of the variable to be displayed in the instance variable contents pane.

Evaluation in the instance variable contents pane: Message expressions typed in the instance variable contents pane can be evaluated.

Evaluation under the scope of self: Any of an inspected object's instance variables can be referenced in the message expressions evaluated in the instance variable contents pane.

Save and update: Expressions evaluated in the instance variable contents pane can be used to modify an object's instance variables; if an expression is "saved," the value of the selected instance variable will be set to the result of evaluating the expression.

Interface with other system tools:
A "method context" Inspector is incorporated into the system debugger. The object inspected in this case is the receiver of the currently selected method in the process walkback list. However, the instance variable list pane of this specialized Inspector displays on the receiver (self), the arguments to the method, and the temporary variables defined in the method.

Inspector functions:
Inspection: Inspection is initiated by sending the inspect message to an object (for example, via an expression typed and evaluated in a Workspace). Inspection can also be initiated by selecting an object and choosing Inspect from the Smalltalk menu. Within the Inspector tool, the menu for the instance variable list pane offers a single function Inspect, which opens a new Inspector on the currently selected instance variable.

Text-editing: The instance variable contents pane provides the standard text-editing menu, supporting the Restore, Copy, Cut, Paste, Show It, Do It, Save, and Next menu functions.

Windowing: The window menu for the Inspector contains seven standard window functions: Color, Label, Collapse, Cycle, Frame, Move and Close.

A key problem with specifications is that the representation is static. The object of design is defined and fixed when it is still just a plan on a piece of paper. Specification ensures properties of the plan—that it is comprehensive, closed; that known problems are addressed; that assumptions are enumerated. But ipso facto it leaves no room to maneuver, no room to explore and invent.

Scenarios as a Focal Design Representation

During the past two decades, scenarios—narrative descriptions and envisionings of people interacting with technology—have become preeminent representations in software design. Of interest, the main historical root emanates from strategic planning, not design studies; perhaps this is the legacy of the new design methods. Also interesting is that the foundations and rationale for scenario-based design, although they are substantively cognitive, do not originate in cognitive science work on problem solving. They largely predate cognitive science and have developed independently of it. Nevertheless, it is interesting to reconstruct cognitive science foundations for scenario-based design. This could guide theory-based development of tools and techniques that might be overlooked on a purely practice-based understanding of scenario-based design. It could also help to produce a more generalized understanding of scenario-based design that might facilitate its application in other design areas.

Scenario-based approaches to strategic planning originated in work carried out in the late 1940s. The best-known example of this work is in Herman Kahn's

Thinking about the Unthinkable. Kahn developed the "accidental war" scenario: An incident of equipment malfunction or unauthorized behavior results in the launch of a single Soviet missile. The missile detonates in Western Europe, and this is immediately detected and disseminated throughout Western military and civilian installations. Although the incident is not interpretable as an attack, the level of anxiety throughout the Strategic Air Command results in one officer's misunderstanding or disobeying orders and firing the missiles under his command. The counterattack is also not interpretable as an all-out first strike; however, it is an escalated military response to the original mistake and could provoke the Soviets to launch their own ready missiles and bombers. In response, the United States might well order the rest of its missiles to be fired, as well as launching its bombers for protection from the Soviet assault and to position them for a subsequent response.

Not a nice story. Indeed, Kahn's planning scenario became a shared nightmare for much of the world for a quarter-century. It was also a very useful design aid. It is still shocking to know that in early 1961 no one had considered planning to communicate with the former Soviet Union during the five to forty minutes it would take for accidentally launched missiles to reach their targets. And yet doing so could possibly have saved the world had the scenario played out. *This possibility was identified by working through the accidental war scenario.* Kahn also developed extensions of this scenario in which the United States and the Soviet Union negotiated a limited nuclear war (this was later explored in the 1960s movies *Fail-Safe* and *Dr. Strangelove*) and in which they unilaterally established a world government (after surveying the first ten million casualties of Armageddon).

The accidental war scenario convinced Herman Kahn that a new regime was needed in strategic planning. He called the scenario a strange aid to thought—"strange" because, despite its shocking revelation, it was not conventionally employed. However, he argued that scenarios provide five distinct advantages to strategic planners.

1. Scenarios help analysts avoid the tempting assumption that circumstances will remain largely the same. They dramatically and persuasively emphasize the wide range of possibilities that must be considered.
2. Scenarios force analysts to address contextual details and temporal dynamics in problems, which they can avoid if they focus only on abstract descriptions.
3. By concretely illustrating a complex space of possibilities and imposing a simple linear rubric of time, scenarios help analysts deal with several aspects of a problem situation simultaneously. They facilitate the comprehension and integration of many interacting elements—psychological, social, political, and military.

4. Scenarios stimulate the imagination, helping analysts to consider contingencies they might otherwise overlook. They vividly illustrate principles or issues that might be overlooked if one considered only actual events.
5. Scenarios can be used to consider alternative outcomes to past and present crises.

The objective of strategic planning is to anticipate and assess future contingencies, resultant situations, and the courses of action they would present. The planner analyzes the problematic nature of current and future situations, situations that are at best only partially understood, and describes and evaluates possible responses. This is actually very similar to design, in which the objective is to envision future artifacts and to assess situations in which those artifacts would be employed in terms of personal and organizational consequences. Essentially strategic planners design courses of action to meet the requirements of future contingencies.

I have slightly extended Kahn's original analysis of scenarios in table 1, more explicitly contrasting the contributions scenarios can make to design to those more properly associated with specifications. Thus, as in the first row of the table, scenarios are deliberately sketchy, tentative, and malleable, whereas specifications are explicit, complete, and final. My point is not merely that one of these is better than the other, but that the two are vastly different and can be expected to make different, perhaps quite complementary, contributions to any design discussion or project.

Table 1

Scenarios versus specifications as design representations

Scenarios	*Specifications*
Sketchy, tentative, malleable	Explicit, complete, final
Breadth-first thinking	Depth-first thinking
Temporal dynamics	Static structures
Raise questions; "what if?" evokes rationale/explanation	Derive/define answers; evokes entailments
Human activity/experience	Information/control flows
Exploits strengths of human problem solving	Controls flaws in human problem solving
Accessible to all stakeholders	Technical, arcane

As shown in table 1, scenarios engage and encourage breadth-first thinking, whereas specifications evoke depth-first thinking. Traditional strategic planning and analysis methods were developed to explore enumerated possibilities, but

they were not effective at identifying new possibilities. Scenarios emphasize temporal dynamics; specifications emphasize structural relationships that are basically static. Scenarios raise questions about why and how a narrative proceeds as it does; they provoke "what if?" thinking about alternative narratives; they cause people to seek and to provide narrative exegeses, explanations for their narratives. Specifications offer definitions and answers; they are designed to settle fundamental questions, not to leave them open or to open them more. The questions they provoke are quite focused questions of logical entailment, that is, the propositions that can be deduced from a given a set of asserted propositions (i.e., the specification). Scenarios depict human activity and experience; they encourage us to identify with the actors described in the narrative and to anthropomorphize the objects of the narrative. Specifications in contrast are about flows of information and control in systems and in user interactions with systems.

Scenarios deliberately seek to evoke empathy, imagery, and meaning making; people are *supposed* to feel something, to see and hear depicted events, to anthropomorphize objects, and to construct their own interpretations. Narratives are important cognitive archetypes of human thinking: we make lessons into folk legends, we dream, we share myths. Specifications are denotative; they definitely are not intended to evoke emotion or other subjective experiences. Indeed, this would be absolutely antithetical to the whole enterprise of specification. Scenarios explicitly seek to leverage the creative characteristics of human problem solving. Not only are people quite good at solving open-ended problems, but they also enjoy working on such problems.

Scenarios present open-ended design problems. In contrast, specifications try to address some of the known flaws in human cognition. For example, people have limited memories, so they often produce internally inconsistent solutions to complex problems. This can happen when they lose track of details, forget assumptions they have made or details of a subsolution, and then go on to address further aspects of a complex problem. Specifications address this by comprehensively codifying problem solutions: indeed, tools and languages for specification ingeniously hide complexity, for example, presenting a top-level view whose components can be successively expanded.

Finally, scenarios are accessible to all stakeholders in a design—users, managers, customers, clients, and even their relatives. A big part of the extraordinary impact of Kahn's accidental war scenario was that everyone could understand it and its significance almost immediately. Specifications, like many technical representations used by professionals, are inherently arcane to nonprofessionals. This is understandable and perhaps necessary, but it is also unfortunate. When all stakeholders cannot participate directly in design, they cannot share their perspectives and knowledge, and the design process itself is undermined (Carroll, *Making Use*).

My first professional position was as a scientist, and later a manager, at IBM. I worked for IBM from the mid-1970s to the mid-1990s. The first functional specification documents I saw were humongous encyclopedias of features and functions (and this was before IBM discovered fonts other than Courier). I noticed that after the culture of human-computer interaction became established in IBM in the early 1980s, interaction scenarios started to appear as appendices to these documents, illustrating how the features and functions specified worked together to produce what we would now call a user experience. By the end of the 1980s, the scenarios had moved up front: The encyclopedic functional specification had become appendix to the sketchy user-oriented scenario. The rhetorical was beginning to lead; the purely functional was beginning to play a supporting role.

Example: A Virtual School

The next two sections illustrate how ostensibly mundane scenario-based practices can be salutary. In the first example, crafting a scenario helped us recognize that we were about to build the wrong system; this is probably the most common mistake in systems design. In the second example, crafting a scenario opened an inclusive communication space for the entire design team, which helped us to allocate roles, creativity, and power more equitably.

When I left industry to become a professor in 1994, I decided to actively explore "alternative" paradigms for system design. I focused much of this effort on projects in the public and civic sector, working directly with teachers and community leaders. I was interested in seeing how scenarios could change software design as I had come to understand it during eighteen years at IBM.

My first major project was carried out under the National Science Foundation's Networking Infrastructure for Education initiative, announced in 1993. This program had a variety of exciting and challenging goals:

Exploit the Internet in public education;
Support collaborative, project-based classroom education;
Increase student achievement with respect to standards;
Improve access of rural schools to new educational opportunities;
Enhance gender equity in science and mathematics education; and
Reduce equipment costs and handling in science education.

This was a highly successful initiative in many ways. The list of goals is a hodgepodge, a wish list of goals, many of which are just as urgent today as they were then.

I led a group at Virginia Tech in proposing a rural infrastructure to connect science classrooms. We argued that this could leverage teachers and equipment across distances, allowing classes to be offered in schools where they rarely could be offered, and enhancing the critical mass of science classes more generally.

Our proposal included a problem scenario describing the challenges that a young girl might experience trying to learn physics given the constraints of her rural school and home. We also developed an envisionment scenario, sketching how some of these challenges might be ameliorated if the girl could have better access to resources, including other students, through the Internet. A shortened version of this scenario is shown here.

Solar System Experiments in a Virtual School

Marissa is a student in Ms. Browning's physical science class at Blacksburg Middle School. The class today was about the solar system. After school Marissa wonders how gravitational relations would work if large gas giants such as Jupiter were much closer to the sun. She accesses Ms. Browning's virtual lab through her Web browser. Randy and David are already there, and she poses her question to them. The three students use a solar system simulation, manipulating various parameters. At the end of the two hours, the students shut down the lab to go on to other work. But before leaving, they agree to log on that evening at seven o'clock to review their findings and write up their report.

This scenario was engaging to National Science Foundation reviewers, as well as to Virginia Tech administrators, who created a remarkable publicity collage depicting it (fig. 7.1) for its new interdisciplinary science center. (I still love this vision: Take an after-school snack and a laptop out to a pasture, somehow pick up a broadband wireless network, and begin exploring the foundations of physics. Perhaps the solar system will just drop in for a visit!)

As we began working with real teachers and students to plan the design and implementation of our virtual school, we experienced directly how scenarios can facilitate critical analysis of whiz-bang technology ideas. It is of course *possible* that a group of students would meet in a virtual environment and engage in revelatory self-initiated investigations of gravitational relationships in solar systems. It is also possible that they would get stuck, or side-tracked, or seriously confused, or just exchange social chitchat. Indeed, since the mid-1990s all of us have seen that the Internet can be a powerful educational medium and resource, but that for teenagers it is often more readily appropriated for social interactions. These are important uses of the Internet, but they are not about doing creative physics.

The teachers and students we worked with on this project actually recognized these issues from the scenario analysis, and they helped our team alter its plan. We became aware of this process through a series of participatory design workshops with teachers in which we analyzed videotaped classroom activities (depicted scenarios of science pedagogy) and deconstructed the causal relationships among the various things students and teachers were doing and the kinds of outcomes they wanted to achieve—and to avoid.

Fig. 7.1 Virginia Tech publicity collage based on the virtual school envisionment scenario

We never implemented the scenario depicted in sidebar 2 and figure 7.1. Instead, we built a highly advanced (for the time) collaborative environment that focused on supporting classroom-to-classroom collaboration, as well as classroom-to-community expert interactions, guided by teachers who helped keep students on track with respect to learning science. Our vision eventually evolved into one of leveraging resources to help make the whole greater than the sum of the parts on a regional basis in high school science education (Isenhour et al.; Carroll et al.). We were fortunate that our original envisionment scenario was not a specification, and we treated it as the starting point for a design discussion.

Example: Community Informatics

A second example, a more recent design study, is the Civic Nexus project, also supported by the National Science Foundation. In this project we tried to understand challenges and design effective and sustainable interventions for technology learning and management in community nonprofit groups, such as local historical societies, food banks, water quality groups, after-school enrichment programs, arts groups, senior citizen groups, regional emergency planning groups, low-income housing groups (for example, Habitat for Humanity), and churches.

The nonprofit sector in American society is fascinating. Although the very name, "nonprofit," emphasizes a sort of economic agnosticism, nonprofit organizations make a huge contribution not only to the economy but also to the social fabric of society, providing social support and social capital on which the coherence of the society depends. Indeed, in the United States much of the social welfare apparatus is implemented through the nonprofit community. Almost 6

percent of all U.S. organizations are nonprofit, accounting for more than 1.6 million organizations and 9.3 percent of all paid employees. The Johns Hopkins Comparative Nonprofit Sector Project estimates that in the late 1990s, in the thirty-five countries worldwide participating in its study, this sector had aggregate expenditures of $1.3 trillion and employed, when factoring in churches, 39.5 million full-time-equivalent workers (Salamon, Sokolowki, and List).

One organization we worked with, a sustainable development group, had an interesting identity problem with its Web site. I was enthusiastic about working with the organization initially because I found the Web site design to be clean and evocative (fig. 7.2). It was easy to read, restrained with respect to blocks of text, and had a graphical environmental theme of green imagery. Indeed, I met my collaborators in this group by browsing Web sites of nonprofit organizations in Centre County, Pennsylvania. I was quite surprised when I finally got to meet these folks face-to-face and learned that they did not want to be perceived as "mere tree huggers."

I learned something from this organization about sustainable development. It is not just a matter of protecting trees, wildlife, and streams; it is about the whole environmental system. It is about balancing economic development with environmental integrity—hence, *sustainable* development. This is, of course, much more ambitious than just restraining economic development and waste in favor of preserving what is natural and clean.

The organization had hired a consultant to rework its Web site. He had misunderstood their identity and mission, much as I had. He had produced a Web site that conveyed this misunderstanding pretty effectively. Indeed, he liked the Web site he produced so much that he refused to change it, and was in effect holding the organization's Web content hostage. When I met them, the organization's members lacked not only the skills to take back control of the Web site design but also the data that their site displayed, for example, water quality data and maps.

Through the course of our collaboration, we emphasized that detailed domain knowledge is a critical asset in design. In this case, knowledge of sustainable development was critical to getting the "right" design. The consultant did not have this knowledge, and although he had solid Web development skills, his knowledge deficit led him to design an attractive but misleading Web site. We helped the staff revalue the importance of their own design knowledge by suggesting that they write scenarios describing the kinds of experiences they wanted their users to have and the way that their Web site information would evoke these experiences.

Doing this literally took just one hour (although fully appropriating a scenario-based practice took a few months). But in just the first week we had a major breakthrough: One member of the organization wrote a scenario in which a local official comes to understand how wastewater management is part of the

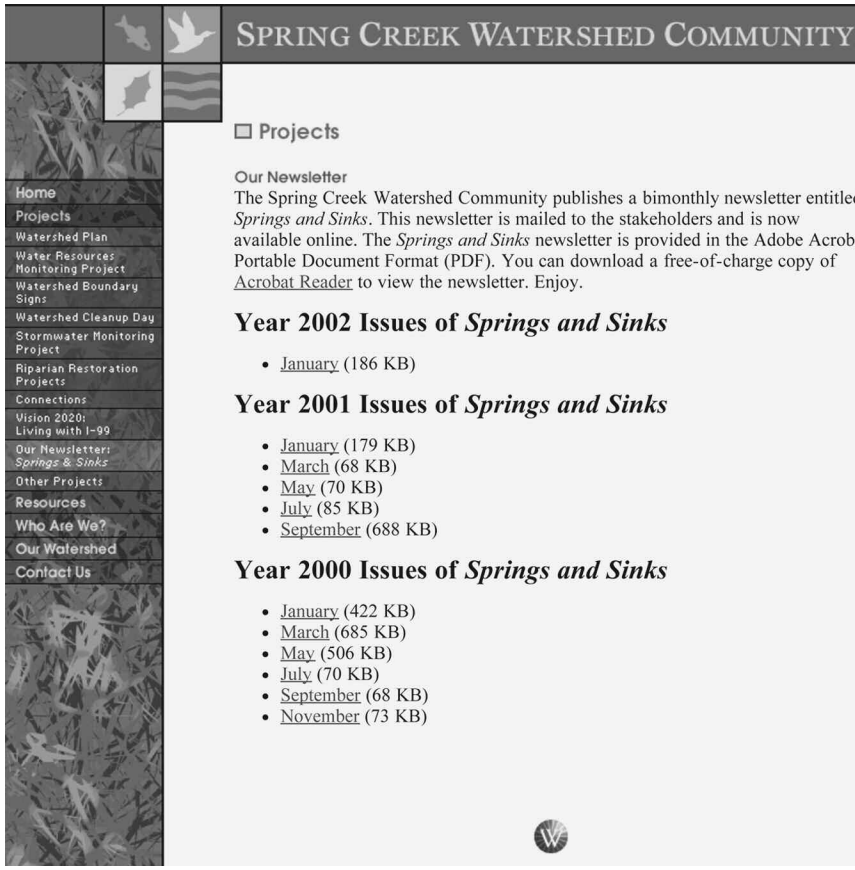

Fig. 7.2 Web site of Spring Creek Watershed Community in 2003

overall environmental system; following is a schematic representation. Prior to this scenario, the group had not focused on the special user group of local political leaders. So this really was an insight, both for Web-site design and for the group's identity in the community. Members of this sustainable development organization could not specify their Web site, but they were easily able to describe it through scenarios, and in fact to elaborate and discover design ideas through scenario envisionment. This allowed them to take control of their Web site through the mastery they already had of their domain and their Web site's content.

Elected Official Design Scenario Created by a Member of Sustainable Development Group

- Newly elected official has heard about SCWC, wonders what is it?
- Googles; browses site

- Pictures of local quality of life, mission statement, strategic goals and stakeholders
- Pursues topic of storm water: damages property, businesses, and local streams
- "Gets it": environmentally responsible development is cost-effective in both the short and long term

This reframing of what design is had a surprising effect with respect to the organization members' earlier attitude of powerlessness with respect to Web programming and development. It seemed that once they convinced themselves that they were the key designers, that their scenarios were the design, they were able to more effectively address the challenge of implementing and managing their design. A few months after the adoption of scenario-based design practices, some members supervised the implementation of a new Web site and then began to personally manage it (Farooq et al.).

Conclusion

I have suggested that, by describing and analyzing information technology designs through narratives of their use, future problems and possibilities can be anticipated and managed. In this sense, such scenarios complement traditional, specification-based design methods, which are strong on detailing static properties of a design but weak on helping designers anticipate problems or achieve possibilities. The virtual school scenario (sidebar 2) and the elected official scenario just discussed are obviously more modest schemes than Herman Khan's accidental war conceptualization. But managing global crises is not the only thing people need to do. These examples show how everyday design breakthroughs are facilitated by scenario-based approaches. Indeed, one point I take from them is that, in the information age, inventing the future may become an everyday task.

Information systems design is in part a software engineering practice. And technical implementations should be specified, if only to create a detailed record of what was done. But the most important business of design always lies in the future, in envisioning and developing possibilities, and in anticipating and managing problems. Rhetoric can be dismissed as "just talk," but scenario-based approaches show how talk is constitutive of design. I believe that architects of technology from all disciplines must learn to narrate the future, and rhetoric obviously has a central role to play in this.

Works Cited

Carroll, John M. *What's in a Name: An Essay in the Psychology of Reference.* New York: W. H. Freeman, 1985.

———. *Making Use: Scenario-Based Design of Human-Computer Interactions.* Cambridge, Mass.: MIT Press, 2000.

Carroll, John M., and Mary Beth Rosson. "Human-Computer Interaction Scenarios as a Design Representation." In *Proceedings of the 23rd Hawaii International Conference on Systems Science*, 555–61. Los Alamitos, Calif.: IEEE Computer Society Press, 1990.

Carroll, John M., et al. "Knowledge Management Support for Teachers." *Educational Technology Research and Development* 51.4 (2003): 42–64.

Farooq, Umer, et al. "Participatory Design as Apprenticeship: Sustainable Watershed Management as a Community Computing Application." In *Proceedings of 38th Hawaii International Conference on Systems Science*, 178. Los Alamitos, Calif.: IEEE Computer Society Press, 2005.

Isenhour, Philip L., et al. "The Virtual School: An Integrated Collaborative Environment for the Classroom." *Educational Technology and Society* 3.3 (2000): 74–86.

Kahn, Herman. *Thinking about the Unthinkable.* New York: Horizon Press, 1962.

Salamon, Lester M., S. Wojciech Sokolowki, and Regina List. *Global Civil Society: An Overview.* Baltimore: Johns Hopkins Center for Civil Society Studies, 2003.