SHAHAF GAL

11

Footholds for Design

In a design environment, knowledge is generated, enacted, and reflected on in an ongoing process between the materials and the participants' intentions and attached meanings. Learning occurs when users create concrete things, such as drawings and models, based on their knowledge, and then reevaluate that knowledge based on what they have just learned.

In conjunction with Donald Schön (see Chapter 9), Shahaf Gal conducted a series of studies of engineering students completing design projects. The story of Ray's bridge, recounted in this chapter, was one of four cases that Gal analyzed in detail. Ray's experience in a 3-week student design competition offers us a distilled and visible version of what happens in more sophisticated and hidden ways in every design project. The problems that Ray grapples with have direct mappings onto software-design contexts: top-down versus bottom-up approaches; the practice of basing new designs on standard examples; the temptation to use materials because they are available; and the struggle between adding features and simplifying.

Ray goes through a directed but winding search, adding and dropping concerns that constrain his design, being surprised by the results of explorations, and making messy decisions that change over the course of the project. He is a student, not an expert, and his lack of sophistication helps to reveal the process more clearly, since the breakdowns are more visible without the facade of expert polished performance. We can clearly see the tradeoffs between different concerns, the abandonment of a path when it seems to have reached a dead end, and the pain of getting stuck. As David Kelley points out in Chapter 8, these phenomena happen to every designer, and are not to be avoided.

—Terry Winograd

COMPUTERS CAN PLAY AN IMPORTANT ROLE in the design process, as *image footholds*. In a design environment, knowledge is generated, enacted, and reflected on in an ongoing process between the materials and the participants' intentions and attached meanings. Learning occurs when users create concrete things, such as drawings and models, based on their knowledge, and then reevaluate that knowledge based on what they have just learned. If we are to respond

Author's acknowledgment: I thank Vanessa DiMauro for bringing the rock-climbing analogy to my attention.

effectively to design situations, our tools—including those based on computers—need to be aligned with the process of the design inquiry.

An analogy for the role of computers in a design environment is that of rock climbing. A rock climber knows that she wants to get to the top of the mountain. She chooses a route for the climb based on her knowledge when she is at the bottom of the hill. As she climbs, she constantly faces new situations where she needs to choose a new footing to proceed. She seeks footholds that are safe and stable on which to pause momentarily to catch her breath and to plan her next step. Each foothold is both an endpoint that sums all the steps she has taken so far, and a point of departure from which to plan the next one. Her choice of a new foothold is determined by the steps that she has taken to get to her current position on the mountain in relation to her goal. She is also guided by her past climbing experience, and she uses her rock-climbing tools as an anchor. At each step, she needs to plan her next step considering what is in her reach; each step presents her with new and different conditions. Thus, a rock climber continuously faces a challenge of making future decisions based on the here and now of her foothold.

Like the rock climber, designers face similar challenges in design settings. Situations of design evolve constantly as a designer secures a point in the design on which he feels safe relying, then moves on again. The following story of a student designing a bridge illustrates the use and value of a computer program that serves as a design foothold. The general principles discussed at the end of the chapter show how the lessons go beyond this case.

The Riddle of *Die Brücke*

Ray was a senior in mechanical engineering at the Massachusetts Institute of Technology (MIT). As a high-school student, he gained experience working with wood, especially with balsa wood. Ray designed and constructed a bridge as part of the fourth annual bridge-design contest at MIT in the winter of 1988. This bridge contest was his first; he worked by himself, in competition with 16 other students from different departments within the School of Engineering. Their

task was to build models of bridges that could best withstand a test load. Participants were given a kit of materials containing strings, wire, glue, and a variety of wood blocks—basswood, balsa, and pine. They had 3 weeks to complete the project.

Students were encouraged to use the engineering laboratory and Growltiger—an expert-system engineering program—to build their models. Growltiger's purpose was to serve students as a design tool and as a virtual laboratory for experimentation in structural engineering. As a virtual laboratory, Growltiger is programmed as a channel of guided discovery, where the system guides designers to enter the necessary data for structural analysis, to manipulate the appropriate computational components, and to evaluate the results against known parameters of structural engineering. The program also assists the designer by providing a default setting that offers standard-sized beams, properties of standard construction materials, and four types of indeterminate structures. Thus, students can test how their designs behave in accordance with a range of standardized degrees of tolerance. The test can serve as a starting point from which students try to optimize the structure. They can then change the shape of the structure, the loading, and the stiffness. Within seconds, the new structure can be analyzed and tested.

Students can also use Growltiger as an open-ended design tool, building their own structures. With the exception of curved supports, students can design the structure, and decide on the kind of material and the loading system; the program will display the deflection. (See Figure 9.2).

Top-Down Design

Ray chose a strategy for building his bridge that attempted to address a range of issues. He strove to manage effectively the time and manufacturing constraints (gluing, amount of work), while creating a strong bridge that was beautiful and original. He aimed to build with a clear design concept. As a partial response to the tradeoff, Ray searched for a bridge with simple structure that would be easy to manufacture. In a design notebook that he kept throughout the contest, Ray wrote at

this early stage:

- Main problem: TIME! TIME! TIME!
- Main constraint: Bridge must be finished in time!
- Main tradeoff: Aesthetics and partial stability versus speed of construction

Ray believed that he could build the bridge in a classical top-down approach: arrive at a concept, prepare a few alternative designs, narrow down to one design, work out its details, check the materials available, and move on to manufacture the bridge.

> What I originally proposed was a clear top-down approach. You start off with comparison of main frames, which means main ideas for designing a bridge, as I did here with several design sketches. For example, take those six ideas of bridges, then take the two best ones, and then elaborate on those to get a good idea of how the geometry looks. Then, go back to the material. From there on, it's a circle among material, manufacturing, and the design; they are all factors from then on, but only from then on. That is a classical top-down approach.

Following this top-down scheme, Ray started by collecting ideas about bridges, and searching for a bridge-design concept. He watched a video of the previous year's bridge contest, looking especially for the mistakes participants made in their design, and he leafed through a book on bridge designs. The Roman arc bridges caught his attention first—they last long, which is proof that they work, he reasoned. Following more reading, he contemplated two other kinds of basic designs—the bow bridge and cable-stayed bridge—and leaned toward the latter in which he had more trust.

After his search, he decided to stick with one general structural idea: to use twin towers—a two-legged bridge—in his design. The concern about settlement of such a bridge introduced the idea of a hinged center span so that each tower would act as an independent element in carrying the load. Given these three considerations, he drew his first design: a two-towered, cable-stayed bridge with a hinged center span (Figure 11.1).

Ray soon realized, however, that the hinge mechanism would destabilize the bridge, and would require much construction work. Still worried about the uneven settlement of the towers, he considered

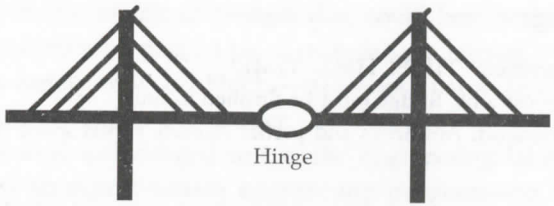


Figure 11.1 Fan Cable-Stayed Bridge with a Center Hinge Working with sketches, Ray could explore both the visual and mechanical properties of his proposed designs. This bridge had a certain elegance, but the hinge could destabilize it.

a bridge with a freely suspended span that would rotate and compensate for the settlement. The bridge would be like an arc, and would swing back and forth to compensate automatically for the settlements (Figure 11.2).

Time constraints and concern about the strength of the strings caused Ray to drop this idea altogether. He decided to tolerate the settlement problem as a constraint, and remained with cable-stayed bridges.

He then tried another angle on bridges, seeking *beautiful* bridges. Hyperbolic bridges and classical cable bridges—such as the Golden Gate Bridge in San Francisco—attracted him the most. In his design notebook, he experimented with cable-stayed bridges arranged in hyperbolic patterns. Ultimately, however, he found this pattern impractical because the nearly horizontal members could carry only small loads.

Feeling the pressure of time, Ray dropped the search for aesthetic bridges and focused on function: cable-stayed bridges with vertical

Figure 11.2 Arc Bridge Each half of the arc could swing back and forth to compensate for settlement. This rough sketch was an attempt to solve a specific technical problem, as a prelude to a more comprehensive design.

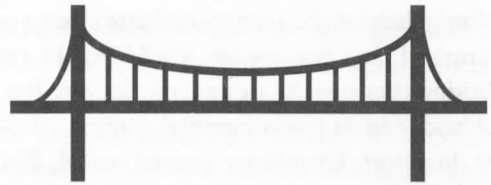


Figure 11.3 Suspension Cable-Stayed Bridge The classical suspension bridge was aesthetically pleasing and made use of the string as a tensile member. In looking for the best overall design, Ray was balancing constraints in several different dimensions, including beauty, strength, and feasibility of construction.

tension members. Ray then placed an additional constraint on his bridge: to use as much as wood and strings as possible, for maximum strength. The kit had plenty of materials, and he liked to include horizontal tension members:

> I am trying to make the maximum with the material that I have been given. The string is the longest piece of material I've been given, and it's a tensile member, so I want to take advantage of it. You could design it without strings, of course: I just believe in using all the material.

The bridge design was then changed into a suspension bridge, as in Figure 11.3.

Ray eliminated this design soon after its inception, because a top cable would weaken the bridge—the bridge needed to be very large and the vertical strings to be very long, and the forces acting on them would be very strong. A few days into the contest, Ray was stuck. Time was becoming a critical factor, and his brainstorm sessions had resulted in neither a clear design nor any elaborate specific structural constraints. Ray decided to change his general approach.

Simulation as a Design Tool

Ray now turned to Growltiger to help him choose from the various designs. His decision to use Growltiger for that purpose did not come easily. After seeing a demonstration of the system, he felt a general mistrust of its capabilities and usefulness. Growltiger, he reasoned, did

not design; it simply helped the user to evaluate design parameters. In addition, he surmised that the system would not be useful for constructing the bridge, because its design model was too simplistic to match an actual bridge in its environmental context.

At this point, however, Growltiger proved useful. Ray prepared six preliminary models of the main bridge types, each reflecting main modes of handling load, and had Growltiger predict their success using default material properties. During the comparison of the designs' general structural behavior, he noticed that two designs showed the greatest strength. He therefore narrowed the possible bridge candidates to two designs: the cable-stayed bridge and the girder bridge.

He then ran a comparative analysis of the two bridges, elaborating and changing their design, and testing various structural options. The bridges, he found, were both strong, but they behaved differently:

> These two performed well. The girder bridge was decent, too, to my surprise. It's the simplicity that gives it a rigidity that's amazing. Looking at how it bends and how it behaves, I could see how stable, it would be—what the overall performance would be.

Ray was using the computer as a medium for reflection, in which to explore the kind of surprises that Schön describes in Chapter 9. His *conversation with the materials* (even in this simulated medium) led to new insights and provided a way to test his specific design ideas.

Conversation with the Materials

After using Growltiger, Ray realized that bridge construction must begin from small components of the bridge. Most important, he realized how critical it was to link the overall design approach with the process of constructing the pieces of the bridge. This realization was the catalyst for another shift in Ray's approach. He changed the direction of his work from top-down to bottom-up, as he became concerned with construction issues, which he had thought he could leave for later. He also began to have second thoughts about building a cable-stayed bridge that relied solely on strings as the main system to transfer the load. The strings, he realized, had limited reliability as

tensile members, especially because the strength of the bridge largely depended on the string attachment to the wood, where the stress will be concentrated. This shift in approach brought about a change in Ray's activities. Ray now needed to test the various components of the bridge. He decided to use the physical laboratory to test the performance of the deck and the supports, which he believed were important to any bridge. He started testing generic examples made of wood.

> When you do experiments, you get an idea of the material. That's something Growltiger doesn't give you at all. You build the box, touch the materials, glue them. I got a lot out of it.

The laboratory tests provided him with a deeper understanding of the structural behavior of the beams and ribs for his deck, which could not be provided by Growltiger's simulation. For a structure of the deck, he compared three alternative approaches. Based on the results, he eliminated two options and decided to go with the box girder, which had unexpectedly proved to be the strongest.

Ray's concern about the weight of the bridge and the use of strings as the main material for the tensiles caused him to recheck the literature on cable-stayed bridges for the kinds of tensiles and cable arrangements used. He learned that the most effective cable arrangement for carrying load keeps the tensiles vertical to the deck. By this point, his bridge design had become detailed and definite: the overall design would be a cable-stayed bridge, with a box-girder deck, two towers, H-supports, and many tensile members made of strings (Figure 11.4).

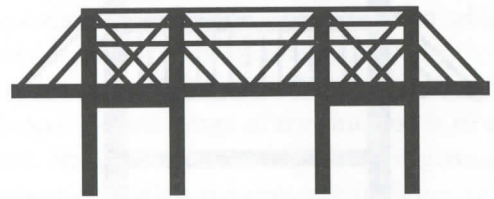


Figure 11.4 Cable-Stayed Bridge with H-Supports and Strings
Ray's experiments with the physical materials led him to consider this design, in place of the earlier suspension design. Working in the laboratory gave Ray insights about practical construction feasibility that were not provided by the computer simulation.

Following his experiments in the laboratory, Ray remained concerned about the use of strings to support the towers. At first, he tried to calculate the load that the strings would carry. He returned to the problem of how to attach the strings, and could not find a solution. At that time, he considered dropping the idea of a cable-stayed bridge. But he could not tolerate the thought of omitting the strings.

> I could not let go 100 feet of worthwhile material! So, I looked for a different solution, and that was the first time that I came up with the idea, "Why not use an additional bar across, and use vertical strings?"

Ray then planned one deck on top of the load-carrying deck, and another below, from which 30 to 40 strings would link the load-carrying deck with the bridge's deck. That design would use the strings and the remaining basswood. This idea also linked and resolved many of his earlier concerns about structural forces, use of strings, and use of as many vertical tensiles as possible.

> I didn't come up with the idea of vertical strings. It came from constraints propagation—from the top and from the bottom. It just suddenly popped up. I stuck with it because I felt good about it. All the problems I was worried about before—the materials, strings that I had, vertical alignment, vertical force performance—were solved by this design.

Ray's bridge design, which he called *Die Brücke* (the Bridge), now looked like Figure 11.5.

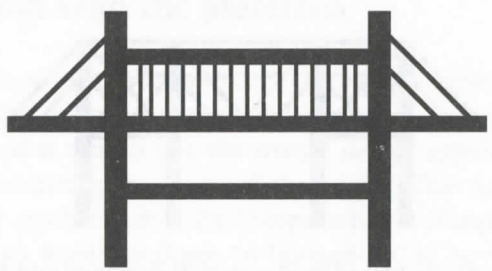


Figure 11.5 Triple-Deck Bridge with Strings as Vertical Tensiles Ray was led to this design in a creative leap (see Kelley's discussion in Chapter 8) that grew out of his persistent focus on how to make use of a single design element—the strings.

Figure 11.6 Ray's Final Design Structural analysis, construction concerns, and a desire to use all of the available materials all played a role in producing the final bridge design.

In the final moments before the contest, Ray decided to add trusses, which he had considered previously, to strengthen the support of the towers. His final bridge is shown in Figure 11.6.

At the testing session that culminated the bridge-design competition, Ray's bridge drew much attention from other participants and from the audience. It passed the first loading test, but failed in the second round, when both ends of the deck were crushed. In the overall contest, the bridge came in fourth place.

Growltiger as an Image Foothold

This detailed case illuminates the many and messy decisions that take place over time in a design situation. The process of design evolves as a process of identifying emerging new questions to address. Through the task, a personal design world is created within which answers are sought out with the use of tools (Figure 11.7).

The personal design world is bounded by the designer's tools and design knowledge. The challenge of the situation is to create a way to test a proposed design solution—which in turn generates new questions. Each point of testing requires the designer to pause on the question, while considering future design questions. She uses design tools, drawings, and models to create design pauses that momentarily freeze the knowledge, and that represent a culmination of the knowledge gathered thus far. Engineering drawings, geometric displays, and algebraic computations assist engineers in maintaining an image, a

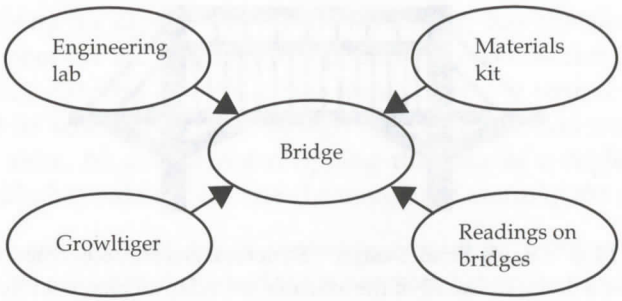


Figure 11.7 Ray's Personal Design World Ray's design world is composed of the tools and design knowledge that he applied to the bridge problem. Each component can lead to new questions and new solutions.

map for orientation, and a language for explaining the design in the process of work.

Reflecting on Ray's experience, it is interesting to observe his use of the computer. Ray used Growltiger to hold images of his design ideas, to test their quality, and to set constraints on his work. And—unintended but perhaps more important—his use of Growltiger shifted his design process. Ray first attempted to work top down. Time constraints and his inability to yield enough concrete constraints on his bridge design caused him to rethink his design approach. When he got stuck, Ray used Growltiger to come up with basic bridge-design ideas that guided his work. From the bridges' simulated performance, he learned about their load mechanisms: the cable-stayed bridge using cables to transfer the load, and the girder bridge using its rigid beam. This knowledge led him to try alterations on each bridge.

The session with Growltiger turned out to be important to his work in another way: It placed the first concrete design reins over his work—from there, he worked within the conceptual frame of these two bridge designs.

Growltiger also served as a preliminary trigger to a moment when—not by intention—all Ray's bridge design theories and work strategies amalgamated into one cohesive image of a bridge. Growltiger provided the critical piece: It was an *image foothold* in the jigsaw puzzle that he was putting together. He used the computer to trigger, *unintentionally*, a critical reflective moment that allowed him

to focus on a bridge design. It also provided him with a reflection of his design strategy. The design strategy and image foothold are part of the same design initiative. They are not easily separated, because it is the strategy that gave birth to the image, and it is the image that informed the strategy.

A significant challenge exists here for software designers. They need to create tools that assist designers to reflect on past steps, and to inform plans for the next foothold—while allowing for the designers' creativity to emerge and to be tested in an *intentional* way.

Suggested Readings

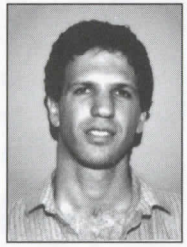
Ken Baynes and Francis Pugh. *The Art of the Engineer*. London: Lund Humphries, 1981.

Shahaf Gal. Building bridges: Design, learning, and the role of computers. *Journal of Machine-Mediated Learning*, 4:4, 1991, 335–375.

Chris Jones. *Essays in Design*. London: Wiley, 1984.

Donald Schön. The design process. In V. Howard (ed.), *Varieties of Thinking*. New York: Routledge, 1990.

About the Author



Shahaf Gal directs the Computers for Instruction Department of the Centre for Educational Technology in Tel Aviv, the largest educational computer research and development company in Israel.

Profile

11. The Spreadsheet

One of the major developments that changed the face of computing was Dan Bricklin's introduction in VisiCalc of the conceptual model of the spreadsheet. Neither VisiCalc's command interface nor its information display (limited by the capacity of the machines that were available, and by how much space VisiCalc required) was remarkable. However, the conceptual model was exceptionally durable. With the additional design work of Mitch Kapor and an implementation on the IBM PC, the successor program, Lotus 1-2-3, quickly surpassed VisiCalc, and was the killer app that moved the microcomputer from the hobbyist's and student's desks into the mainstream of the business world. In its time, it was a radically new idea; it led to the PC revolution that populated millions of offices all around the world with desktop computers.

In hindsight, the idea of a computer spreadsheet seems obvious. The accountant's ledger sheet, with its two-dimensional grid of rows and columns, has long been a fundamental tool in professional accounting practice. When an accountant prepares a budget or financial statement, typically each row represents a different line item, and each column presents the amount of that item in a given time period. There were many programs on mainframe computers, before VisiCalc, that produced outputs that looked like ledger sheets. But a dramatic (and largely unexpected) change came from the spreadsheet's interactivity, and from the role that interactive use played in the activity of financial modeling.

Forecasting is a basic business activity, which calls for projecting alternative financial results based on choices and assumptions, such as the expected sales, cost of goods, possibilities for investment, and so on. Results, such as the amount of profit or loss, are calculated for

Profile Author: Terry Winograd

each of the alternatives, often for a sequence of time periods (month by month, or quarter by quarter for several years). In practice, preparing a projection or a budget is an iterative design process. Different sets of assumptions lead to different results. On the basis of seeing these results, the financial analyst often wants to see what would happen if assumptions or strategies were changed. Doing multiple revisions on paper requires laborious reentry and recalculation. Programmable calculators sped up the calculation aspect, but still left users with much furious button punching and number scribbling.

Bricklin's insight was that the financial-projection process could done interactively on a microcomputer. He created a computer screen that mimicked the structure of the paper ledger sheet (Figure 11.8) so that it would be familiar and easily adopted. He extended the possibil-

B31: (,3) [W11] +B20-B29 READY
Line Bar XY Stacked-Bar Pie
Bar Graph

	A	B	C	D	E	F
15	Profit and Loss Statement					
16						
17	(in Millions)	---------Actual--------			- Projected -	
18		1984	1985	1986	1987	1988
19						
20	Revenues	3.865	4.992	5.803	6.022	6.481
21						
22	Expenses					
23	Salaries	0.285	0.337	0.506	0.617	0.705
24	Utilities	0.178	0.303	0.384	0.419	0.551
25	Materials	1.004	1.782	2.046	2.273	2.119
26	Administration	0.281	0.288	0.315	0.368	0.415
27	Other	0.455	0.541	0.674	0.772	0.783
28						
29	Total Expenses	2.203	3.251	3.925	4.449	4.573
30						
31	Profit (Loss)	1.662	1.741	1.878	1.573	1.908
32						
33						

16-May-87 1:15 PM

FIGURE 11.8 The Spreadsheet The now-familiar format of a spreadsheet was developed for VisiCalc and refined in Lotus 1-2-3, shown here. The traditional arrangement of rows and columns in an accountant's ledger was the base for a new kind of affordance—the ability to calculate changes in an entry automatically, based on a formula that defined the entry's value in terms of other entries. Although the idea seems simple, it revolutionized the way that financial work is done. (*Source:* Courtesy of Lotus Development Corp.)

ities for the contents of an individual cell, to include not just text or a number, but also a formula for calculation, which can be based on results from other cells. With this structure, all the logic necessary to recalculate the spreadsheet is stored in the spreadsheet. When the contents of a cell are changed in a spreadsheet, all the cells whose values depend on that one are automatically, almost magically, changed. All the complexities of intermediate calculations are invisible, unless the user chooses to examine them.

The spreadsheet was fundamentally different from earlier programs for financial calculation, with their unbridgeable separation between program and data—corresponding to a nearly unbridgeable separation between programmer and accountant. The key innovation was not in the interface elements or the software structure, but rather in the *virtuality* provided to the user—the underlying world of objects and their behaviors. The spreadsheet virtuality combines the regular structure of the familiar ledger sheet with an underlying structure of interlinked formulas. The nontechnical user can build a complex financial model incrementally, and can explore the model through successive iterations of inputs. This quantitative change in ease meant a qualitative change in how people worked with the data.

The interactivity of the spreadsheet made it possible to create what Shahaf Gal (Chapter 11) calls *design footholds*. The person designing a financial plan can quickly represent alternatives and explore the consequences of specific decisions. The power of the spreadsheet lies in the interaction between calculating results and inventing new possibilities.

An additional dimension that appeared in the next generation of spreadsheets, initiated by Lotus 1-2-3, was the ability to write short macros that could reproduce the action of a series of key strokes. Further, by the inclusion of simple control constructs (if–then, go to, etc.) and interfaces to the user interface (to display and process menus and prompts), the macro capability provided a general capability for user programming. In a way, user-created macros were the solution to the tension between direct manipulation, with its direct mapping of action to result, and programming, with its use of abstractions to create patterns of activity that do not depend on the specific data. The initial spreadsheet moved away from the programming-based models of mainframe financial software, making it highly usable but limited in power. The addition of macros brought back a good deal of that pro-

gramming power to the ordinary user, or at least to the *superuser* (or local expert) whose background was in the financial world, but who could produce macros for use by other users (see Nardi, 1993).

Now that the spreadsheet is widely available, it has come to be used for many tasks that have nothing to do with finances. A spreadsheet can be used for any activity that calls for calculating regular arrays of values that are interrelated by regular formulas—especially for those activities that call for exploring alternatives. Professors use spreadsheets for grading courses, scientists use spreadsheets for interpreting data from experiments, and builders use spreadsheets for keeping track of materials. New kinds of spreadsheets have been developed that fill the cells with visual images, sounds, and other data representations, interlinked by formulas that perform calculations in the appropriate domain.

The lessons to be learned from this history are not about the specifics of the spreadsheet; they are about the underlying reasons for its power.

The power of representations. Although the underlying calculations for financial modeling were not new, the representation of an active array of formula-based values created a new virtuality—a world in which to work.

The power of interactive modification. Because the recalculation of a spreadsheet could be done interactively as part of the flow of a modeling process, it could be used as a design foothold—as a way of making concretely visible a set of assumptions and relations, seeing what they produced, and using the results to guide the next round of modifications.

The power of incremental programmability. The macro language for spreadsheets created a vast army of superusers, who did not see themselves as programmers, but who could produce spreadsheet templates that carried out complex and useful work for themselves and for colleagues in their workplaces. Giving end users control of their tools was a major theme of the PC revolution. When users became less dependent on a priestly caste of programmers to accomplish their tasks, their productivity flourished.

Suggested Reading

Bonnie Nardi. *A Small Matter of Programming: Perspectives on End User Computing.* Cambridge, MA: MIT Press, 1993.