Pioneers and settlers: Methods used in successful user interface design

Article · June 1995

CITATIONS
22

READS
201

1 author:

Stuart K. Card
Stanford University
206 PUBLICATIONS **30,714** CITATIONS

SEE PROFILE

Pioneers and Settlers: Methods Used in Successful User Interface Design

STUART K. CARD
Xerox Palo Alto Research Center

The pioneers get the arrows, and the settlers get the land.

GUY KAWASAKI
(former Apple evangelist, BayCHI meeting,
Palo Alto, California, April 14, 1992)

User interfaces, it has been said (Card, this volume), are an ineluctable part of interactive software systems. They are typically more than half the code, often far more. They typically cause more than half the problems, too—often far more. In short, they represent the sort of troublesome engineering problem that organizations would like to *do something* about.

This book is one of those attempts to do something. The premise of this section of the book is that by examining successful user interfaces of the past, we can spot and articulate methods that led to successful designs and then deploy these methods in the future, leading to more successful designs. This stratagem, of course, presupposes we know success, at least when we see it. Unfortunately, the world is not so simple. Many spreadsheet programs contain demonstrably successful user interfaces. Moreover, they were designed using a successful, reliable, repeatable method—simple theft. Are such programs examples of what we mean by successful interface design and the methods for producing them? In a sense, as we shall see, the answer is yes. But also no. Despite its obvious virtues, theft has the equally obvious limitation that the method works only on things previously created by some other

method (and for a similar purpose). It also has the equally obvious limitation, observable in spreadsheet programs, that technological advance slows to a crawl and all systems begin to look alike.

In what follows, we attempt to understand what it means for a user interface to be successful. Perhaps it will be no surprise that this will cause us to expand the unit of analysis beyond a single design point to its historical roots and to its arena of commercial or mission engagement. In particular, it will suggest the difference between "Pioneer Systems," those aimed at advancing the state of the art and "Settler Systems," those aimed at exploiting existing techniques. Armed with a notion of success, we then set about to inventory the methods used by the designs described in other papers of this section. Finally, we attempt to taxonomize methods in our inventory, organizing them so as to gain insight into the structure of the space of methods available for building interactive computing systems. We suggest that below the surface of the methods there are a set of common mileposts or goals. That these mileposts be accomplished in some way is probably more important than the particular methods of accomplishing them.

In the small, success of HCI systems is focused on the next system being built. But in the large, success has to do with the whole process of technology evolution.

Success

Successful User Interfaces

Let us begin with the hard question: What do we mean by a successful user interface design? Or perhaps better stated: What do we mean by a successful design for an interactive computing system? In one sense, it is not problematic to find such designs. Both the popular press and informal professional discussions list them frequently. This is fortunate since identifying the methods underlying successful designs would be much harder in a field where it was generally thought that there weren't any. Table 1 is my (incomplete) list of successful interfaces. This list, like any, is a bit idiosyncratic, but it would be surprising if there weren't considerable overlap with the lists of others.

For each of the systems in Table 1, we can give a reason for listing it as a success. The first groups of these develop the notion of the electronic workspace:

Sketchpad (Sutherland, 1963) originated the concepts of virtual reality and constraint-based interfaces and most of the rest of computer graphics. The

AAMRL Virtual Reality System (Furness, 1986) developed practical versions of virtual reality, especially in the cockpit context. The NASA Virtual Reality System (Fisher, 1989) produced cost-reduced components for virtual reality and led to its popularization.

Along the path that eventually led to the GUI (graphical user interface) paradigm, NLS (Engelbart and English, 1968) was one of the major precursors to direct manipulation. It introduced the mouse, hypertext, the point-and-click style of text editing, and groupware. Expert users could work collaboratively at unbelievably rapid rates. Smalltalk (Kay, 1977; Kay and Goldberg, 1977; Smith, 1977; Tesler, 1981) took the mouse from NLS; further developed the window concept into overlapping windows; and added menus, objects from Simula (Dahl and Nygaard, 1966), and the desktop metaphor. The result was the invention of the direct manipulation interface and the principled integration of different media, such as text, graphics, music, and animation. Star (Smith et al., 1982; Johnson et al., 1989; Miller and Johnson, this volume) added the notions of the menu bar, generic functions whose interpretation depended on the object to which they were applied, and strong integration. It was the first commercial expression of the desktop–icon interface. The achievement of Star was that a noncomputer professional could begin operating a high-functionality mixed-media interface in about 30 minutes. The Macintosh interface (really the Lisa interface, Williams, 1983) was the first commercially successful exploitation of this new sort of interface. It added user interface commonality across third-party applications and dialog boxes. The achievement of the Macintosh was that it provided a mixed-media environment in which there was high transfer of learning among third-party applications.

Dataland (Herot, 1980; Bolt, 1984) explored the notion of the interface being a window into a large data space. Rooms chopped this space into regions organized around tasks, in order to get rapid task switching in a large space without search, and allowed objects to be simultaneously in different regions.

The next groups of user interfaces go beyond the usual workstation:

FreeStyle was a groupware system that originated the use of coordinated audio and gesture annotation to image-based documents. PenPoint (Carr and Shafer, 1991) introduced the pen-based notebook metaphor.

The X protocol (Scheifler, Gettys, and Newman, 1988), though not itself a user interface, is notable because X-based user interfaces swept user interaction in the Unix industry into a client–server model, thereby allowing for user interfaces whose backends ran on distributed machines.

The Olympic Message System (Gould et al., 1987) developed techniques that allowed multicultural, multilingual, walkup users of an information kiosk to exchange messages and access information. It also demonstrated the

TABLE 1 Some Successful HCI Interfaces

• Sketchpad	Virtual reality and constraint-based interfaces
• AAMRL Virtual Reality	Virtual reality
• NASA Virtual Reality	Cost-reduced virtual reality
• NLS	Mouse, hypertext, point and click, groupware
• Smalltalk	Overlapping windows, menus, objects, icons, desktop metaphor
• Star	Menu bar, generic functions, integrated functions, large-scale UI
• Mac	Third-party commonality
• Dataland	Large desktop data space
• Rooms	Multiple shared workspace
• FreeStyle	Image-based synchronized documents
• PenPoint	Pen-based integrated UI environment
• X	Client–server model applied to user interfaces
• Olympic Message System	Multilingual kiosk interaction with thousands of walkup users
• Emacs	User-extensible editor
• Unix shell scripts	User-extensible operating system
• VisiCalc	Spreadsheet metaphor
• HyperCard	User programmability

feasibility and value of user prototyping, even under the most demanding schedule constraints.

Finally, a last group of user interfaces developed the notion of user-tailorable systems:

Emacs (Stallman, 1987) and Unix Shell Scripts (Kaare, 1983) were both successful attempts to make extensible systems that are user programmable and tailorable. They have proved powerful and highly adaptable over a number of years despite their lack of either aesthetics or human factors. VisiCalc originated the spreadsheet—a paradigm so powerful that it overcame human factors shortcomings in the command language.

HyperCard (Apple Computer, 1987) brought a version of hypertext and frame-based interface (Robertson, Newell, and Ramakrishna, 1981) to the mass market, provided good media integration for sound, CD-ROM, video, and animation, and in particular, was successful at simplifying programming for at least some end users.

To this list of systems, we can add the examples of systems discussed as successful systems in other papers in this book (Table 2). For these systems, we have specific information on the methods by which they were produced. Two of these, Star (Miller and Johnson, this volume) and FreeStyle (Francik, this volume) also appear in Table 1 and have been mentioned. The system described by Atwood, Gray, and John (this volume) is a proposed toll-and-assistance operator workstation (we call it the New TAO Workstation). Their claim is not for the success of the workstation, but of their method, CPM-GOMS, for evaluating it. This is one of the first studies in HCI to document commercial return for using an analytical method; in this case they claim to have saved the phone company more than $2 million. Rally (Wixon and Jones, this volume) is a fourth-generation application generator. Wixon and Jones discuss the revision of this system's existing user interface. Their claim is that their methods had large and measurable effects on the product's commercial success. Whiteside presents a user-customized meeting room and training system, custom-fit to each user (we call it Meetingware).[1] The claim is that the methods described led to a new class of products.

For all the systems in Table 1 and Table 2, there is some reason, as we have outlined, for counting the system a success. But looking at these systems as a collection, success seems to be something of a mixed bag. Some were important commercially. Some were hailed by reviews of the time and received awards, but were commercially unsuccessful. Some were known only to academic specialists, but were heavily imitated. Clearly, creative success and business success are not the same. We need to distinguish the varieties of success that are possible before we can discuss the methods underlying that

TABLE 2 Successful HCI Systems as Represented at the Workshop

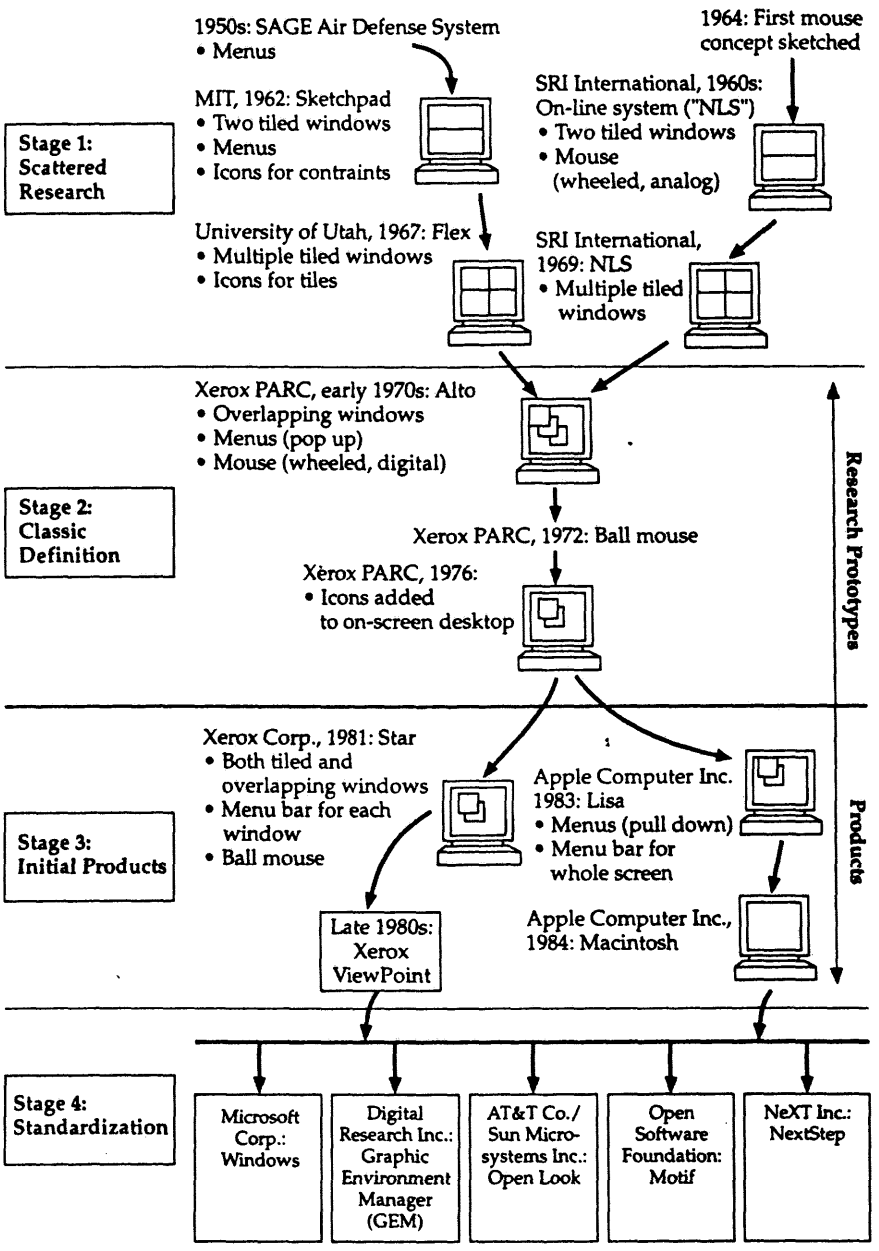
Paper	System	Description	SSI
Atwood, Gray, and John	New TAO Workstation	Proposed toll-and-assistance operator workstation	1
	CPM-GOMS	Analysis into GOMS methods and critical path method analysis	(4)
Wixon and Jones	Rally	Fourth-generation application generator	1
Whiteside, as presented at the workshop	Meetingware	Custom meeting room and training	3
Francik	FreeStyle	Pen-based groupware	6
Miller and Johnson	Star	Networked document processing	8

success. There is also a more subtle lesson to be learned. In a sense, focusing on point designs, such as those listed in Tables 1 and 2, is the wrong unit of analysis. Actually, the methods that produce successful, especially novel, user interfaces can best be seen by expanding the context and examining the *sequence* of inventions that led to those designs—the *food chain of innovation.*

The Food Chain of Innovation

Figure 1 gives a reasonably accepted account of the history of the graphical user interface as compiled by the editors of *IEEE Spectrum* (Perry and Voelcker, 1989). Several of the systems in Table 1 appear in the figure. From the figure, we can see that development of the graphical user interface went through four stages: (1) scattered research at various university and government-financed laboratories, (2) the classical definition of this interface at Xerox PARC, from which all commercial versions descend, (3) the introduction of initial products into the marketplace, and (4) standardization. Those systems that were commercial successes are marked in the figure. An obvious conclusion stands out: *Commercial success occurred only at the very last stages of this process, long after major technical invention had ceased.*

FIGURE 1



Development of the graphical user interface according to the editors of *IEEE Spectrum* (Perry and Voelcker, 1989). The interface went from scattered exploratory research, to development of a series of classic systems, to initial product, to standardization. The graphical user interface is now an entrenched incumbent technology, resistant to innovation.

For example, consider the Apple Macintosh, widely cited as the first commercially successful use of a graphical user interface. The first Apple introduction of this technology on the Lisa failed, as did the second, the Lisa 2, as did the third, the Macintosh 128. Only on the fourth try, the Macintosh 512, was there commercial success. But this machine had no user interface invention, it just used the design settled earlier in the series. Most of the real invention in this design, in turn, actually occurred in the designs of the Xerox Smalltalk and the Xerox Star systems and related design at Xerox PARC, especially Bravo (Lampson, 1976), Gypsy (Lampson, 1988), Draw (Lampson, 1976), and SuperPaint (SIGGRAPH, 1990). Some ideas in these systems can be traced even further, as Figure 1 shows. In fact, it should be noted, a major set of the precursor ideas derive from government-funded university and research institute laboratories or major industry research laboratories. Virtually all the user interface invention occurred in the predecessors to systems that were commercial successes.

The Macintosh 512 was a commercial user interface success, but it was not an invention success. Smalltalk and Star, and perhaps Lisa, were invention successes, but they were not commercial successes.[2] These invention successes were of dominating importance in making possible the success of later user interface designs, but they may have occurred in previous research or in a previous commercial system. Furthermore, many of them worked well in the sense of allowing users to accomplish the task the system was built to do—that is, they were also engineering successes. *If we were simply to examine a system that was a commercial success, we would miss many of the key methods that gave rise to that success because they occurred earlier in the food chain.*

It is useful to distinguish at least three types of successes in user interface design: (1) an *invention success,* (2) an *engineering success,* and (3) a *commercial innovation success.* An *invention success* brings into existence something that did not previously exist. It usually relieves or avoids some major constraint inherent in the previous art, or it contrives a way to perform some new function. Inventions can be things, like a computer input device, or they can be processes, like a technique for testing user interfaces. Closely related is a *discovery,* which brings something into known existence that previously existed, but was unknown. An *engineering success* brings into existence a system that meets its objectives and does so using a specified level of resources. An *innovation success* brings a new product or service to the market and derives commercial gain from so doing.

In adopting these terms, we have followed the conventional distinction between *invention,* the creative act of bringing new things into existence, and *innovation,* the particular new marketable product or process or service that eventually results (see, for example, Burgleman and Maidique, 1988, p. 31). The one is the beginning, the other the end, of the innovation process. As

anyone with any experience with this process has learned, there's a lot that goes in the middle.

Invention

The success of an invention may be assessed by the extent to which it made a difference—its impact. Of course we know this largely in retrospect—the extent to which the impact of some invention shook the existing state of the art like an earthquake, overturning existing practice. In fact, earthquakes are such a good metaphor we can adopt the phenomenological scaling of earthquakes literally and scale inventions the same way. Newell (1992) used this technique to invent the NOV scale for expressing the novelty of a theoretical prediction. Each level on the NOV scale, from 1 (a reformulation) to 3 (a confirmation of existing theory) to 6 (a new discovery) and beyond indexed successively greater impacts of a theory on the existing state of knowledge. In a similar vein, Table 3 proposes a new Seismic Scale of Innovation (SSI). The table pairs terms from the phenomenological Modified Mercali earthquake magnitude scale to describe successively greater impacts of the invention on the existing state of the art. Since it is a phenomenological scale, the Modified Mercali scale is pretty good at describing metaphorically just about anything whose foundations are being shaken. For convenience, equivalents to the more familiar Richter scale (an objective scale based on ground shaking) are used to set numerical scale points. An important characteristic of the earthquake scale is the relative frequency with which the different scale events occur: Smaller scale events occur frequently, and great events on rare occasions. I have therefore given a very rough guess of relative frequency using the exponential distribution of the Richter scale and anchoring the end points so as to produce the 50,000 programs that are said to have come into existence at the one end and the field-transforming development of the GUI at the other end. Of course, such approximations are necessarily crude, but they do produce a feel for relative invention in HCI.

A Class 0 Invention on the SSI scale ("No earthquake") is a program that just clones another without modification; the clones of VisiCalc come to mind. Then come SSI Class 1 Inventions ("Not felt, but recorded"), which make minor modifications to existing user interfaces or user interface techniques in order to handle similar problems. Microsoft Word for the PC was essentially BravoX for the Alto (Lampson, 1988), for example. In a Class 2 Invention ("Hanging objects swing"), known techniques are applied to new problems. PowerPoint, a program for making presentations that

TABLE 3 • Seismic Scale of Innovation

Earthquakes			User Interfaces		
PHENOMENA	MM	RICHTER	SSI	CHARACTERISTICS	N/YR
No earthquake.	0	0	0	Clone (Example: VisiCalc clones)	—
Not felt, but recorded.	I	0–1.9	1.0	Reimplementation of existing technique to similar problem (Examples: Microsoft Word, New TAO Workstation, Rally)	4000
Hanging objects swing.	II–III	2.0–3.4	2.0	Application of existing technique to new problem or minor additions to technique (Example: PowerPoint)	830
Felt by some. Like a passing light truck.	III	3.5–4.2	3.0	Nonobvious use of existing techniques (Examples: Cosmic Osmo, Meetingware)	170
Felt by many. Dishes rattle.	IV	4.3–4.8	4.0	Refinement of existing paradigms by substantial new invention (Example: Macromind Director)	36
Felt by all. Sleepers awake.	V–VI	4.9–5.4	5.0	New idea for minor component (Examples: Dataland, Rooms)	7.5
Slight building damage. Books fall. Liquids spill. Windows break. Walking is difficult.	VI–VII	5.5–6.1	6.0	Discovery of new major component (Examples: Rocky's Boots, Emacs, FreeStyle)	1.6
Considerable building damage. Chimneys fall. Some houses knocked from foundations.	VIII	6.2–6.9	6.5	Trendsetter. Imitated by others (Examples: Spreadsheet, Shells, HyperCard, Macintosh UI)	0.66
Serious damage. Rails are bent. General panic. Partial collapse.	IX	7.0–7.4	7.0	Course change for industry (Example: X)	0.32
Great damage. Masonry buildings destroyed. Bridges fall.	X	7.4–7.9	7.5	Major paradigm shift (Examples: NLS, PenPoint)	0.13
Damage nearly total. Most works of construction destroyed.	XI–XII	8.0	8.0	Restructuring of field (Examples: Sketchpad, Smalltalk, Star)	0.067

combines a drawing tool with templates, color palettes, and a slide sorter, might be an example here. The vast number of user interfaces produced are Class 0, 1, or 2—their intention is to use known techniques in a new product or service.

In the next band of levels, the user interface begins to receive some notice for inventiveness. In a Class 3 Invention ("Felt by some. Like a passing light truck"), nonobvious use is made of existing techniques. Cosmic Osmo's (Miller, Miller, and Lovick, 1989) clever creation of a world that can be explored without instruction by three-year-olds might be an example here. In a Class 4 Invention ("Felt by many. Dishes rattle"), there is refinement of existing paradigms by substantial new invention. The set of techniques used in the Olympic Message System (Gould et al., 1987) might qualify here. In a Class 5 Invention ("Felt by all. Sleepers awake"), there is a new idea for some component of the user interface. An example would be the large data space in the MIT Dataland system (Bolt, 1984). In a Class 6 Invention ("Slight building damage....Walking is difficult"), there is the discovery of some major new component. Emacs's (Stallman, 1987) extensible editor paradigm might be an example.

In the last band, there is substantial impact on the field of user interface design. In a Class 6.5 Invention ("Considerable building damage.... Some houses knocked from foundations"), a user interface emerges that is a trendsetter, widely imitated by others, for example, spreadsheets from VisiCalc. In a Class 7 Invention, there is a course change for at least part of the industry. In fact, the gloss for this level ("Serious damage. Rails are bent. General panic. Partial collapse") is a pretty good description of the impact of X (Sheifler, Gettys, and Newman, 1988). In a Class 7.5 Invention ("Great damage.... Bridges fall"), there is a major paradigm shift. The NLS system, which introduced the mouse, point-and-click editing, and hypertext is an example. Finally, in a Class 8 Invention ("Damage nearly total.... Most works of construction destroyed"), there is a restructuring of the field. Fifteen years later, most new user interfaces do bear a resemblance to those pioneered by the Smalltalk system.

The systems indexed by Table 3 seem to fall into two broad groups that we will call Pioneer Systems and Settler Systems. In one group (SSI $\geq$ 3), the Pioneer Systems, are Star, HyperCard, PenPoint, and NLS, user interface designs that pioneered new user interface techniques or uses. In the other group (SSI < 3), the Settler Systems, are interfaces that apply existing interface techniques to new problems. The systems of the first group are concerned to a greater extent with user interface invention. The systems of the latter group are concerned mostly with exploiting existing techniques for other ends, hence with good engineering. (Note, however, that all user interfaces must be concerned with engineering to some extent, or even good inventions will fail in the implementation.) In Table 2, Star, FreeStyle, and Meetingware are examples of Pioneer Systems. The New TAO Workstation and Rally are examples of Settler Systems.

Engineering

Whereas the success of an *invention* may be measured by its impact, the success of an *engineering* project may be assessed by the extent to which it meets its objectives (and by whether those objectives are adequate). Engineering is about having enough control over technique that a desired objective can be predictably and reliably achieved for a given resource cost. Usability is an important engineering objective. In the spirit of the SSI scale, I also offer a validated scale for usability: the *Cooper–Harper Scale* (Table 4), drawn from the testing of airplanes (Cooper and Harper, 1969). Since it is a phenomenological scale, the Cooper-Harper is also pretty good at describing just about any machine where usability is important in order to get it to fly. The user is assumed to be able to compensate for some of the deficiencies of the machine, and the scale essentially indexes how hard the user must work to achieve a well-defined level of success with the system (not crashing). The 10-point scale breaks into four regions: Satisfactory without improvement (1–3), Deficiencies warrant improvement (4–6), Adequate performance not attainable without improvement (7–9), and Not controllable (10).

It is not so useful to try to rate the systems in Tables 1 and 2 because rather than looking at a single system, engineering progress would usually be seen within *versions* of the same system. For the Rally system, however, this is, in fact, what we have. Redesign improved the system from perhaps a 6 ("Very objectionable but tolerable deficiencies requiring extensive user compensation") to perhaps a 2 ("Good—Negligible deficiencies, user compensation not required"). For the New TAO Workstation system, the principle engineering metric of improvement was time per call. The CPM-GOMS method predicted that, to everyone's surprise, this metric was expected actually to worsen. In both cases, it is not the invention of novel functionality that is at issue; it is better performance on usability or efficiency for the defined system.

Table 3 suggests that most user interfaces are Settler Systems, dominated by engineering (96 percent of the systems will be SSI 0, 1, or 2, according to the table) and that therefore methods for engineering, including usability engineering, will be of most interest to most designers. On the other hand, it also suggests that methods that aid invention will have a disproportionately large impact.

Innovation

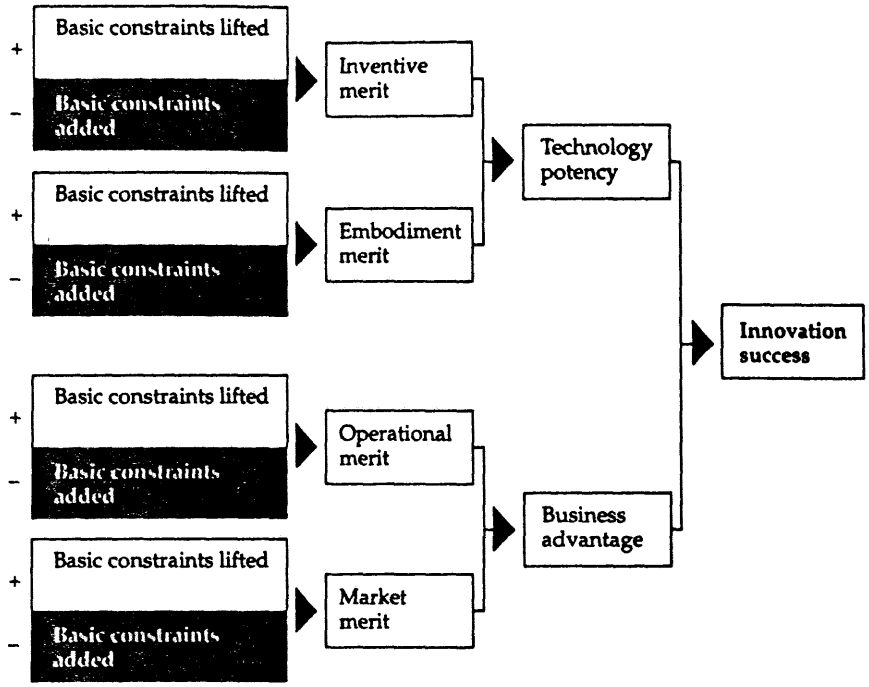
But what of the other ingredients that lead from invention and engineering to commercial success? What makes this other part of the difference? Here it is

TABLE 4 **Cooper–Harper Scale**

Score	Aircraft Characteristics	Demand on Pilot
SATISFACTORY WITHOUT IMPROVEMENT		
1	Excellent—highly desirable	
2	Good—negligible deficiencies	No pilot compensation required
3	Fair—some mildly unpleasant deficiencies	Minimal pilot compensation required
DEFICIENCIES WARRANT IMPROVEMENT		
4	Minor but annoying deficiencies	Moderate pilot compensation
5	Moderately objectionable deficiencies	Considerable pilot compensation
6	Very objectionable, but tolerable deficiencies	Extensive pilot compensation
ADEQUATE PERFORMANCE NOT ATTAINABLE WITHOUT IMPROVEMENT		
7	Major deficiency	Adequate performance not attainable with maximum tolerable pilot compensation. Controllability not a question
8	Major deficiency	Considerable pilot compensation required for control
9	Major deficiency	Intense pilot compensation required to control
NOT CONTROLLABLE		
10	Major deficiencies	Control will be lost during some portion of required operation

useful to broaden our view and look at user interfaces as just another class of technological innovation. White and Graham (1978) have suggested that four factors determine the success of a technical innovation. These are inventive merit, embodiment merit, operational merit, and market merit (Figure 2). *Inventive merit* is the technological invention that makes a difference, as we have previously discussed. Usually the invention relieves or avoids major constraints inherent in the previous art. In the case of the transistor (Table 5), the electronic amplification function could be provided with smaller size,

FIGURE 2



White and Graham's (1978) model of innovation success. Success depends on the extent to which innovations exhibit four kinds of merit. Innovations often fail in the market because they are lacking merit in one or more of these categories, despite strengths in other areas.

TABLE 5 **Innovation Criteria for Transistor Radio**

1. Inventive merit	+	Improves size, weight, power, reliability
2. Embodiment merit	+	Miniaturization of antenna, tuning capacity, batteries, loudspeaker permits pocket radio
3. Operational merit	+	Removes need for franchised dealer service network and dealer channels because of higher reliability and can ship through the mail
4. Market merit	+	Captures major new growth market with go-anywhere, play-as-you-go characteristics

After White and Graham, 1978.

less power, and more reliability. Of course an invention may have disadvantages, too. Transistors had low yield, causing them to be more costly than vacuum tubes. So it is important, as in Figure 2, to keep track of those factors detracting from innovation success (marked with a minus in Figure 2) as well as those contributing to innovation success (marked with a plus).

Embodiment merit refers to the way the invention is embodied in its surrounding systems. An embodiment can either leverage off the opportunity of an invention or dissipate it. Japanese manufacturers leveraged the size and power advantages of the transistor by miniaturizing the antenna, the speaker, and the capacitors as well. This made possible the transistor radio, a radio with new uses. On the contrary, American manufacturers just made slightly smaller radios and thereby dissipated the invention merit of the transistor in their embodiment.

Operational merit is how the invention affects the business practices of the company selling it. Again, the operations of a business can leverage off the advantages of an invention or dissipate them. The small size and greater reliability of transistor radios made it possible for transistor radio manufacturers to avoid having to establish a franchised dealer and service network. The radios didn't need as much service, and they could be mailed to a central facility. This allowed the manufacturers to overcome a major barrier to the entry of new companies into the radio business.

Finally, *market merit* is how the invention is leveraged to increase demand. The small, lightweight transistor radios could be carried in a pocket. This allowed new uses for the radio: It could be carried to the beach or to a ballgame; people could listen to sports or music while they did construction work or washed the car. These new uses expanded the market even though transistor radios were at a cost disadvantage. The new markets' expanded volumes helped to lead to lower prices, and lower prices expanded the markets further.

We can use White and Graham's analysis to help us understand the relationship between success in technological innovation for HCI and commercial innovation success. Table 6 shows the case of the Xerox Star, the system that first commercially introduced graphical user interfaces. The Star system had spectacularly strong technological merit. It introduced windows, generic functions, menus, and personal networking, all the basic ingredients that would become the standard user interface ten years later. Its embodiment, on the other hand, was more mixed. Experience had shown that larger screens were much more desirable for productivity, and marketing analyses had concluded that price was less important than functionality, so the machine was given a larger display, faster network, and integrated applications, but these raised the price. The heavy emphasis on the machine as a network citizen meant that it did not work well without the net and hence was expensive in

TABLE 6 Innovation Criteria for Xerox Star

1. Inventive merit	+	Windows (multiple processes share display). Generic functions (fewer commands, faster learning). Menus (recognition instead of recall). Networking (group interaction). More function, less learning
2. Embodiment merit	+	Larger screen, faster Ethernet, integrated applications for higher productivity
	–	Embodiment led to high entry cost, blocked third-party software. Underpowered
	–	Not intended to look like general computer
3. Operational merit	–	System product not well-matched to Xerox sales, service force. High evolution cost
4. Market merit	+	Opened new markets. Strong customer loyalty
	–	Embodiment means must sell against IBM in MIS department. No cheap entry. Can't develop enough software

small installations. The decision not to sell it as a general-purpose computer or to enable third-party development meant that its uses could not grow as fast as those machines that did this. Finally, the machine was not fast enough to leverage the productivity gains inherent in its inventive merit.

Operationally, the Star was not well matched to the capabilities of the sales and service force. The heavily integrated design led to high evolution costs for the machine. Still, the machine had strong market merits. It opened new markets and developed a very loyal customer base. But the embodiment, which was aimed at selling to large corporations rather than individuals, meant that it was forced to sell against IBM in MIS departments dominated by IBM. The lack of adequate third-party development meant that others could not open new applications for it. Thus, the Star system was a strong technical success for HCI, but not a commercial innovation success.

The Apple Macintosh (Table 7) presents a different story. The Macintosh had more modest inventive merit. The essentials of the graphical user interface came from Smalltalk and Star, to which were added a dialog box scheme and a scheme of generic pull-down menus that eventually allowed transfer of learning by users from one application to another. Apple embodied the technology in a general-purpose computer at a low price that was open to third-party developers. This was an important leveraging of the technological merit. In fact, this orientation toward third-party software was probably

deciding in the end. It was a third-party software company, not Apple itself, that found the "killer-ap" of desktop publishing, which turned the invention merit of the bitmapped graphical interface into a major advantage for the machine. On the other hand, there were also negatives to the embodiment: The machine was underpowered, the screen was too small, there was no serious network, and there was little integration. Operationally, the machine was compatible with the company's marketing and service channels, which was a plus. Apple was already a computer company with a dealer network. But it was in the marketing merit that Apple was able to derive from the technical invention that really helped bring success. The interface gave the machine market differentiation from all PC companies. Ease of use became a major selling point. The machine attracted innovative third-party developers and opened up the desktop publishing market and desktop graphics design markets.

Wang FreeStyle (Table 8) is still another case. FreeStyle also had good inventive merit. It allowed users to attach synchronized speech and gestures as annotations to image-based electronic mail, and the whole was embodied in an image-based desktop. The basic innovation capability was partially enhanced by its embodiment. The electronic mail embodiment allowed a new kind of collaboration at a distance. Synchronized interactions that would normally have required a high-bandwidth video conference installation ("Why is this number here lower than this number over here?") could be done with low bandwidth over existing networks with no synchronous requirement. The image desktop part of the embodiment allowed integration of scanning, fax,

TABLE 7 Innovation Criteria for Apple Macintosh

1. Inventive merit	+	(From Smalltalk and Star): Graphic orientation, windows, menus, mouse
	+	Generic pull-down menus allow transfer of learning among third-party applications
2. Embodiment merit	+	General-purpose computer at a cheap price
	–	Screen too small, underpowered, no network, lack of integration
3. Operational merit	+	Compatible with channels
4. Market merit	+	Market differentiation from PC. Ease of use major selling point. Positioned as the "other PC." Attracted innovative third-party developers. Opened up desktop publishing

TABLE 8 Innovation Criteria for Wang FreeStyle

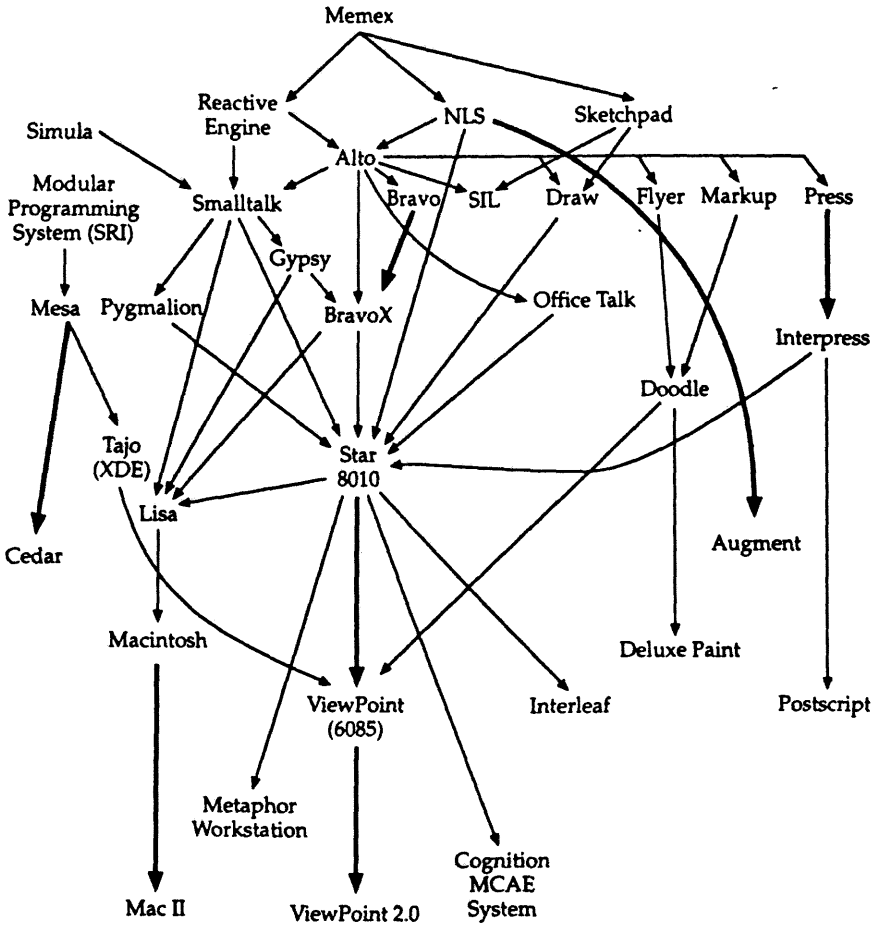
1. Inventive merit	+	Synchronous speech acts in asynchronous medium: synchronized speech, gesture annotations to image-based electronic mail
2. Embodiment merit	+	Electronic mail embodiment allows new kind of collaboration at a distance, but with lower bandwidth, existing nets, no synchronous requirement. Image desktop allows integration of scanning, fax, drawing, image capture from PC applications
	–	Constrained by Wang proprietary systems
3. Operational merit	–	Not compatible with Wang channels, products, PC products
4. Market merit	+	Market differentiation, new market development
	–	Confusion about what product is and for whom

drawing, and image capture from PC applications. Alas, some of the inventive merit was dissipated by the requirement that it had to be embedded in Wang proprietary systems. Operationally, FreeStyle was not well-matched or compatible with Wang channels, products, or with PC products. This fact did not allow Wang to leverage the invention well. Market merit was also mixed. On the one hand, it was a market differentiator and opened up new areas for potential market expansion. On the other hand, there was considerable confusion about what the product was and for whom it was meant.

It is easy to see why invention success and commercial innovation success are very separate things. Even starting with a strong invention, it may take several tries before the right embodiment, operational use, and market positioning of an invention are hit upon. Worse, generally only invention success and embodiment success are assessable from within the research and development organizations. Even the best set of methods in the hands of the best people can therefore fail commercially.

To summarize, by expanding our focus beyond individual point design first to the food chain of ideas, then to the larger innovation context, we can now finally state more about what it means for an interactive system to be successful: *An interactive system design is successful if it is an **invention success** (that is, it relieves major constraints inherent in previous art or captures some new function) or if it is an **engineering success** (that is, it accomplishes its design objectives, such as usability or speed, within its resource constraints) or if it is an **innovation success** (that is, it produces a return on investment, namely, it has an adequate*

FIGURE 3



History of the Star user interface (Miller and Johnson, this volume). Star benefited from the food chain of ideas and, in turn, was a very strong contributor to later systems.

combination of technological merit, embodiment merit operational merit, and market merit) or if it is a ***link on the food chain*** *to the above.*

An invention's success is usually shown by its being widely imitated, often in modified form, over generations of systems. This is clearly seen in the diagrams that are often drawn tracing the food chain of invention ideas. Figure 1 is one such diagram. Figure 3 is another. Each step is a stage in the evolution of the technology. A successful invention *makes a difference* in the history of the technology. The systems listed in Figures 1 and 3 have achieved some measure of success by virtue of appearing in these diagrams: They led to something.

An engineering success is shown by the fit that is achieved to its purpose. Unlike invention success, which is best seen across generations of the technology, engineering success occurs within the design process of a single system, possibly over several versions. It is usually achieved by some combination of analytical methods (that predict the parameters of a design that will fit the situation) and the empirical test–fix cycle that studies the design in idealized or realistic environments to discover areas of misfit and plan remedies.

An innovation success is shown by the extent to which the organizational success of the sponsoring organization is affected by the introduction of the innovation. In the commercial world, this is in the form of profit return on investment, or perhaps strategic positioning or some other subtle surrogate for profit. In the noncommercial world, the analogue might be in terms of organizational growth or in terms of mission success (for example, battles won, disease deaths reduced). An innovation success is a successful engagement with the larger context surrounding an invention.

Methods

We have seen that different sorts of success can be associated with user interface design. Design methods are somewhat relative to these. Methods that support engineering, for example, might not be suitable for invention. Methods are also relative to different aspects of the invention process itself. We therefore continue our analysis of the food chain of invention by considering briefly the nature of the invention process that gives rise to it.

Methods and the Invention Process

Studies in the history of technology help us see patterns underlying technological evolution. Hughes (1983) studied the activities of professional inventors whose inventions resulted in the electric power industry. He identified five phases of invention: (1) problem identification, (2) solution as idea, (3) research, (4) development, and (5) introduction. In *problem identification*, the inventor perceives a situation that can be defined as a problem. This implies that a solution is likely to be found. Identifying a problem in a situation is an important perception. Experienced inventors realize that in many situations, problems cannot be defined because of the inadequate state of technology or for nontechnical reasons. *Solution as idea* refers to the idealized functioning of some solution in some idealized environment. The idea is often represented

as a sketch. *Research* is an information-gathering exercise done by literature search or experiments, possibly with prototypes. *Development* may involve the redefinition of the problem and new ideas (and perhaps additional research). The invention is tried in environments increasingly similar to the one in which the innovation must function. Finally, *introduction* brings the invention to market or otherwise to real users. Of course, this process is neither straightforward nor linear. Still Hughes's phases are useful in helping us recognize that innovation is an activity of parts, not a single homogeneous act.

Hughes's phases are applied to the invention process for the papers in this section (Table 2) in Table 9. The New TAO Workstation, for example, is at the point of introduction. The analysis methods described are trials determining whether it should be introduced or not. The Rally system is concerned with introduction but also with engineering out problems of usability. This is done by trying to understand and appreciate the use of the system in the environment in which it must actually function. Both these systems (at least in this phase of their development) are Settler Systems. They are primarily concerned with engineering performance and usability using known user interface techniques on defined problems.

By contrast, the other three systems in Table 2 are Pioneer Systems. In Table 9, this is reflected by the number of phases with which the systems are involved. The FreeStyle system and the Star system are more concerned with the invention of new technology for a new problem. The Meetingware system is concerned with integrating existing technology to solve a new problem; the invention here is a method for running meetings, eventually introduced as consulting service business.

TABLE 9 **Phases of Invention (Hughes) Applied to Systems in Table 2**

	Settler Systems		Pioneer Systems		
	[1]	[2]	[3]	[4]	[5]
SSI	1.0	1.0	3.0	6.0	8.0
Problem identification				●	●
Solution as idea				●	●
Research			●	●	●
Development		●	●	●	●
Introduction	●	●	●	●	●

[1] New TAO Workstation, [2] Rally, [3] Meetingware, [4] FreeStyle, [5] Star

FIGURE 4

An interpretation of the nature of invention according to Hughes (1983). An invention is a technology that uses some resource efficiently to meet a demand precisely.

Let us consider more closely the problem identification phase. Of course, invention may follow many courses and have many origins, but Hughes noticed some frequent patterns in the methods used by professional inventors that are useful to us. These inventors often sought to identify a demand or function to be performed on the one hand and a resource on the other (Figure 4). Then they would try to develop new technology that used the resource efficiently to meet the demand precisely. If the demand is stipulated and the search is for technology to fill it, then the invention is "demand pull." If the technology is stipulated and the search is for a demand, then it is "technology push." Both exist and can lead to successful systems.

According to Hughes, inventors identified a problem by looking for a "reverse salient," a weakness in the technology. They then tried to identify a "critical problem," that is, a problem to be solved that would correct the reverse salient. For Edison, the reverse salient was usually economic, the critical problem usually technical, thus linking the parameters of invention success to some of those for innovation success.

What is important to realize is that the process of bringing out an innovation may focus on either of these activities: searching for a new function ("application" in the computer industry) that utilizes a technology or searching for a technology that is a good fit to a defined problem.

Now we turn to the five systems of Table 2 in order to discover the methods used in their design. The phases of problem as idea, research, and development can intermingle, and their disentangling requires more detailed information than is available to us from the reports we have of their development. To simplify matters, we will distinguish the more easily separated methods surrounding artifact construction and evaluation. Introduction, as

we have seen from White and Graham's analysis earlier, is a very complex topic in its own right and one whose methods are not dealt with in detail by these accounts. We therefore consider it outside the scope of our current analysis. As a consequence, we are left with three aspects of the design process:

1. *Problem identification*
2. *Artifact construction*
3. *Evaluation*

We use these to give initial organization to the inventory of methods we discover in the subject systems of our analysis.

New TAO Workstation (Atwood, Gray, and John)

In the design of the New TAO Workstation, the functional demand has been stipulated as minimizing time to service a call. Atwood, Gray, and John are evaluating the system on this basis. They use several methods: The *Comparative Field Experiment* is used for a comparison between the existing and proposed workstations. CPM-GOMS (*Analytic Calculation*) is used to predict the performance of the new system and to find a hidden bottleneck that is the cause of the differences. To do the calculation, *Scenarios* are developed from videotaped analysis of operators doing benchmark tasks (*Benchmark Laboratory Experiment*). The analysis is also used to interpret and reinforce the results of an empirical experiment that would otherwise be uninterpretable. We can summarize their methods categorized by design aspects as follows:

METHODS USED:

Problem Identification

- *Scenarios*

Evaluation

- *Benchmark Laboratory Experiment*
- *Comparative Field Experiment*
- *Analytic Calculation: GOMS*

Rally (Wixon and Jones)

In this system, again, the functional demand has already been established. Wixon and Jones are called in to engineer an improvement in usability. They use a large number of methods. First, they use some evaluation methods at the site of use to locate problems in fit between the uses of the system in

context and its design: They do *Field Interviews* with users at their site of work and *User Observation* of users attempting to use the original system. Then they use another set of methods for constructing a prototype of the artifact: Users are encouraged to make suggestions and to evaluate the prototypes (*Participatory Design*). The designers use their own experience with other systems (*Steal and Modify*) for ideas as well as their own experience as users (*Use What You Build*). They set up some *Agreed Formal Usability Metrics* and design to these as targets (usability engineering). For their design, they make use of a form of *Object and Action Analysis* and use *Generic Functions* to reduce the command count and simplify the design. The design is iterated (*Cut and Try*), and iterations are evaluated with *Field Interviews, User Observations, Benchmark Laboratory Experiments* to measure performance time, *Comparative Field Experiments* to compare versions, and *Analytic Calculations* to evaluate designs on the basis of number of menu transitions. *Comparative User Preferences* between original and proposed design is used as a wide-band technique to pick up unanticipated problems, especially among the important expert user population. Finally, usability problems are put into the problem tracking system with the rest of the engineering problems (*Usability Problem Tracking*).

METHODS USED:

Artifact Construction

- *Steal and Modify*
- *Cut and Try*
- *Participatory Design*
- *Formal Usability Targets (UE)*
- *Usability Problem Tracking*
- *Object/Action analysis*
- *Generic Functions*

Evaluation

- *Field Interviews (videotaped)*
- *User Observation (videotaped)*
- *Benchmark Laboratory Experiments (Time)*
- *Analytic Calculations* (Number of menu transitions)
- *Comparative User Preferences*

Meetingware (Whiteside)

This is a case of demand pull. Whiteside starts from a relatively fixed notion of functional demand: use of emerging computer and communications technologies to produce better synchronous, real-time meetings both local and geographically distributed. His search is for better meeting methods and

technologies. His group used their own actual meetings (*Design for Yourself*) with many iterations (*Cut and Try*) to discover techniques for improving meetings and to try out the technologies. They also invited other people to their design meetings and went to meetings of others (*Participatory Design*) and used *Special Projects* to expand the meetings beyond the group. Building was mainly *Cut and Try* systems integration. Evaluation was by *Use What You Build.*

The project has several good examples of finding and resolving reverse salients: The need for a portable, high-resolution color projector is identified as a bottleneck and is developed through encouragement of a third-party manufacturer; system integration is identified as another bottleneck blocking the development of meeting room tools. Finally, the role of new meeting methods is identified as an enabling factor for integrating the technology, and this is addressed. In fact, its identification shows that instead of a simple software or hardware business, a successful business model is likely to come as part of a custom consulting business.

METHODS USED:

Problem Identification

- *Design for Yourself*
- *Cut and Try*
- *Use What You Build*
- *Participatory Design*
- *Special Projects*

Artifact Building

- *Cut and Try*
- *Participatory Design*
- *Special Projects*

Evaluation

- *Use What You Build*
- *Special Projects*

FreeStyle (Francik)

This is a case of technology push.[3] The technologies (image capture, voice recording, electronic mail, fax, pen annotation, high-resolution graphics) were established early based on a very generic notion of usefulness; the development attempts both to engineer these to acceptable levels of performance and to search for a more precise notion of demand. To develop a

prototype, the designers use a version of the *Design for Yourself* method in which a particular user or very few users are selected and the design made to satisfy them (within the group the slogan was, "Design for Leonard," the manager) through repeated iterations (*Cut and Try*). The voice and gesture annotations are integrated with fax documents and screen images of computer-generated documents. A principal operation on these documents is a technique for combined handwriting, voice, and gesture annotation. With this technique, a number of group work interactions that would have required high-bandwidth synchronous communications can now be done with low-bandwidth asynchronous communications. But the emphasis this brings to group communications makes it necessary to search for and understand the parameters of demand.

The Wang team had a double burden. They needed good human engineering to make the system sufficiently paperlike that the concept did not die on issues of system response (for example, a tablet with a "bad feel" relative to paper), and they needed to reshape the feature-set of the system to bring such added value to its users that it could bring a new computing paradigm into the marketplace. The team used many methods for these tasks. They used the *Try to Break It* method to reveal some shortcomings in their test of candidate tablets. *Benchmark Laboratory Experiments* were used to test the tablet and the whole system. Among other things, task time, error frequencies, and comments were collected. *Questionnaires* were used in the design of the hardware. *User Preferences* were assessed using 7-point *ratings* of important attributes (such as pen thickness), and *rankings* were done among alternative designs.

Parallel to the laboratory tests, a number of field methods were used with both *Alpha Testing* in house and *Beta Testing* outside. The team and their colleagues used the system (*Use What You Build*). An extensive set of *Field Interviews* and *Field User Observations* were made at different sites. *Participatory Design* was undertaken with users, and "Communication Constellation Analysis," a variant on Sociogram Analysis, was done to identify communication patterns (*Work Flow Analysis*). To describe possible uses uncovered in their field studies, the team wrote brochures describing possible system applications, a form of *Scenario.*

METHODS USED:

Problem Identification

- *Field Interviews*
- *User Observation*
- *Scenarios*
- *Design for Yourself*

- *Cut and Try*
- *Use What You Build*
- *Work Flow Analysis* (Sociogram)
- *Participatory Design*

Artifact Building

- *Steal and Modify*
- *Cut and Try*
- *Participatory Design*

Evaluation

- *Benchmark Laboratory Experiments*
- *Alpha Testing*
- *Beta Testing*
- *Try to Break It*
- *Use What You Build*
- *User Observation*
- *Questionnaires*
- *Comparative User Preferences* (Rankings)
- *Absolute User Preferences* (Rankings)

Star (Miller and Johnson)

Star was a technology push effort.[4] It arose not so much from a particular demand as from a vision whose pedigree extended from Vanevar Bush, through Doug Engelbart and Ivan Sutherland, Alan Kay, and others. This led to a long series of prototypes conducted by the *Design for Yourself* and *Use What You Build* methods in which an entire community of people experienced several thousand person-years in an environment of networked, bitmapped, personal computing. The results were several new genres of software: point-and-click editing (derived from NLS), bitmap-based paint programs, geometric primitive-based drawing programs, laser printing, client–server networking, page description languages, work-flow-based computing, object-oriented graphical user interfaces, icons, pull-down menus, and the desktop metaphor. Problem identification took the form of integrating this experience, formal market analyses, and designer fieldwork. In addition to the *Use What You Build* prototyping method, the directions of the PARC technology and Star were partially guided by an extensive market analysis in which Xerox employees did extensive *Field Interviews* and analyses with about 60 businesses, including a *Work Flow Analysis* for each. This was followed by

designed mockups of technical solutions to customer problems, which were brought back to customers for their reaction, including queries about pricing. One result is that it was concluded that customers were relatively price insensitive and were more interested in functionality. As a consequence, the Star design point improved usability (for example, going to a larger display) at the expense of price. The Star interface designers then went to customer sites, preselected by the market analysis, conducted *Field Interviews* and *Work Flow Analysis*, and wrote *Scenarios*.

As Miller and Johnson illustrate, the Star designers had a large amount of accumulated exposure and experience going into the design, and this was heavily used (*Steal and Modify*). It is this experience that provided grounding for the larger ideas (which probably couldn't have been tested experimentally, anyway). The fundamental technique for getting the design clean was to produce a set of abstractions that captured the essence of user work using *Objects and Actions Analysis* (Newman and Sproull, 1973). The methods of *Design Before Coding* and *Comprehensive Design Description* were techniques for achieving global consistency in the design. This was appropriate because the extensive prototyping effort preceding the Star design had already provided the experience needed for this step. From there, *Cut and Try* was used to prototype very rapidly and to try out design variants that the group was undecided about or to validate that some idea was likely to work. Only if it was not obvious from looking at the prototypes were *Comparative Laboratory Benchmark Experiments* necessary. In some cases, *Theoretical Calculations* were used to supplement or confirm the experiments. For example, *Benchmark Laboratory Experiments* compared novice users on several alternative methods for determining the optimum number of buttons on the mouse for text selection, but *Analytical Calculations* were used to compute likely expert performance (since the design was new, experts didn't, of course, exist to run in experiments). In another case, the maximum velocity that the user would move the mouse was calculated (Card, Moran, and Newell, 1983, pp. 252–255) and found to be greater than the hardware could support, forcing a redesign. (The calculation was done in a discussion during lunch and a confirming experiment completed by the early afternoon.) Usability problems were put into the prioritized fix-it list with other system problems (*Usability Problem Tracking*), and a *Try to Break It* method was used as part of the release process. But one of the principal methods for building the interface was *UI Constraints First*, that is, the user interface constraints were set before other constraints of the system, forcing other subsystems to adapt. This was partially enabled by the method of *Designer Management*, in which the user interface designer is also the product manager and has clear control of the design.

METHODS USED:

Problem Identification

- *Design for Yourself*
- *Use What You Build*
- *Scenarios*
- *Work Flow Analysis*
- *Field Interviews*
- *User Reaction to System Mockups*

Artifact Building

- *Steal and Modify*
- *Objects and Actions Analysis*
- *Generic Functions*
- *Comprehensive Design Description*
- *Design Before Coding*
- *Cut and Try*
- *UI Constraints First*
- *Usability Problem Tracking*
- *Designer Management*

Evaluation

- *Alpha Testing*
- *Use What You Build*
- *Benchmark Laboratory Experiments*
- *Try to Break It*
- *Analytical Calculations*

The Ecology of Methods

Patterns of Method Use

Now that we have analyzed each of the systems in Table 2 to discover the methods used in each, we can note some larger patterns of use. Table 10 summarizes the methods used for each of the systems of Table 2, broken out by whether they were used in Problem Identification, Artifact Creation, or Evaluation. There are several patterns to note. First, the same method can appear in several roles. *Field Interviews*, for example, were used in both problem identification and evaluation.

Second, methods can be classified into either *analytic methods* (understanding is extracted by analysis) or *synthetic methods* (understanding is extracted by building something). These are marked A or S in the table. Both analytic and synthetic methods were used in all three roles, problem identification, abstract creation, and evaluation. In developing the FreeStyle system, for example, the demand and functionality to be performed was investigated by building prototypes and using them, a synthetic method (*Use What You Build*). But the same sort of information was also sought by *Field Interviews*, an analytic method. The different methods are presumably complementary. Several projects used suites of methods with similar aims to gather deeper insights and correct for any misleading impressions gained from a particular method.

Finally, it is interesting to note that the combinations of methods used were more eclectic than ideological. For example, on the Rally project, methods associated with the "contextualist" school were used (for example, *Participatory Design, Field Interviews, User Observation*), but so were *Benchmark Laboratory Experiments* and *Analytic Calculations* that are sometimes considered part of an opposing philosophy. Similarly, the Star project, often associated with *Benchmark Laboratory Experiments* (Bewley et al., 1983) and sometimes with *Analytic Calculations*, also used extensive *Field Interviews and User Observation*. Whiteside discusses the contextualist approach at length in his paper on Meetingware, but a major part of the designers' demand identification was done using themselves as subjects rather than using external field groups, as might be thought from the philosophy. The FreeStyle system used all flavors of methods. Even the New TAO Workstation group, primarily interested in *Analytic Calculations* with GOMS models, used *User Observation*.

Although methods from supposedly opposing philosophies were used together, I detect no contradictions here. The designers and evaluators of these systems are simply sophisticated enough to employ methods within their limitations. The Rally group, for example, used contextualist methods to understand the design problem but were able to use *Analytic Calculations* to search the design space cheaply for promising menu designs. They then used more expensive contextualist methods to evaluate their design. Likewise, the Meetingware group used themselves to iterate their understanding as rapidly and cheaply as possible but then looked at other groups and used *Special Projects* to validate and generalize their findings. Rather than thinking of methods as ideological opponents of each other, these groups use combinations of methods to complement each other and cheaper methods to conserve resources so that the more expensive methods could be deployed in the most vital places.

TABLE 10 Summary of Methods Used

Method	Type*	New TAO Workstation	Rally	Meeting System	FreeStyle	Star	Total
PROBLEM/DEMAND IDENTIFICATION							
Design for Yourself	S			●	●	●	3
Use What You Build	S			●	●	●	3
Scenarios	A	●			●	●	3
Participatory Design	S			●	●		2
Field Interviews	A				●	●	2
Work Flow Analysis	A				●	●	2
Cut and Try	S			●	●		2
User Observations	A				●		1
Special Projects	S			●			1
User Reaction to Mockups	S					●	1
ARTIFACT CREATION							
Cut and Try	S		●	●	●	●	4
Steal and Modify	S		●		●	●	3
Participatory Design	S		●	●	●		3
Object and Actions Analysis	A		●			●	2
Generic Functions	S		●			●	2
Usability Problem Tracking	S		●			●	2
Formal Usability Targets	A		●				1
Special Projects	S			●			1
Design Before Coding	S					●	1
Comprehensive Design Description	S					●	1
UI Constraints First	S					●	1
Designer Management	S					●	1
EVALUATION							
Benchmark Laboratory Experiments	A	●	●		●	●	4
Use What You Build	S		●	●	●	●	4

Method	Type*	New TAO Workstation	Rally	Meeting System	FreeStyle	Star	Total
Analytic Calculations	A	●	●			●	3
Field Interviews	A		●		●		2
User Observation	A		●		●		2
Comparative User Preferences (rankings)	A		●		●		2
Alpha Testing	S				●	●	2
Try to Break It	S				●	●	2
Comparative Field Experiments	A	●	●				2
Special Projects	S			●			1
Beta Testing	S				●		1
Questionnaires	A				●		1
Absolute User Preferences (attribute ratings)	A				●		1

*A = analytical method, S = synthetic method

It is interesting to note the most popular methods. If we restrict ourselves to methods used by a majority of the projects (or more projects), the chief methods for problem and demand identification were:

Design for Yourself

Use What You Build

Field Interview

Scenarios

The most popular methods for artifact creation were:

Steal and Modify

Cut and Try

Participatory Design

Thus, to come back to a question we began with: Is simple theft an example of what we mean by a successful method? The answer according to these projects is clearly, yes. The reason is that, as we have noted, technology advances by the food chain of ideas, so good designers are also designers knowledgeable about the designs of other systems, and there is probably at least as much

power in this method as in participatory design or any of the methods aimed at contextual understanding of the problem to be solved. But although theft is the engine that carries good ideas forward in the chain, other methods are necessary in order to add to the stock of good ideas.

The most popular evaluation methods were:

Benchmark Laboratory Experiments

Use What You Build

Analytic Calculations

Table 11 breaks out the methods used by Analysis vs. Synthesis and phase of the design process. This table shows that about 5 of the 23 methods (about a fifth) are used in more than one role. It also shows the heavy use of *Cut and Try* and *Use What You Build*, two of the more pragmatic methods.

Dimensions of Variation

The eclecticism of methods displayed by designers of these real systems suggests looking more closely at the morphological space (Card, Mackinlay, and Robertson, 1991) occupied by the methods. We can describe the space of methods on at least four dimensions (Figure 5):

1. *Analytic vs. Synthetic Methods*
2. *Idealization of Context*
3. *Idealization of Representation*
4. *Development Milepost*

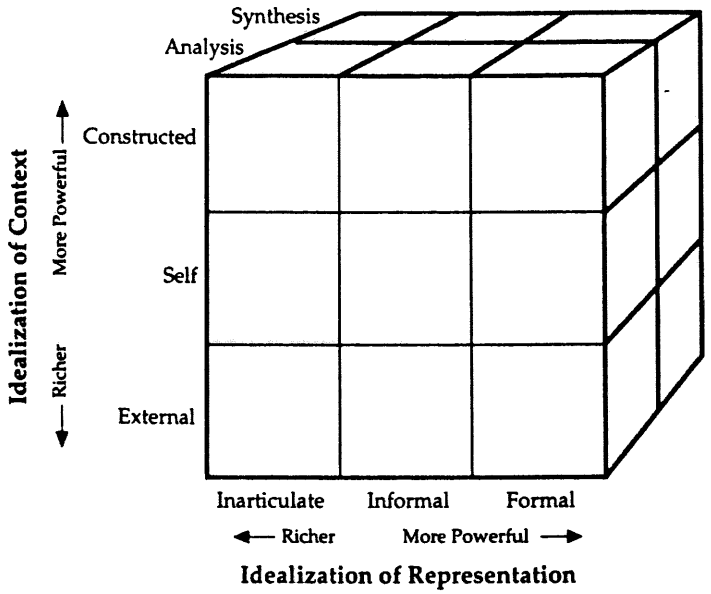
First is the distinction between *Analytic Methods* and *Synthetic Methods* that we referred to earlier. Second is the amount of *Idealization of Context* involved. At the one extreme are methods in which there is little idealization of context. The methods are aimed at the direct external setting in which a system will actually be used. *Field Interviews* and *User Observation* are examples. These methods attempt to tap the richly textured, contextualized information that comes from the natural setting without any intermediate abstraction or encoding. A less expensive alternative to dealing directly with an external cost is the use of the self or local group as a context. Methods like *Use What You Build* and *Alpha Testing* attempt to exploit this lower-cost resource. At the other extreme is a constructed, artificial context. *Benchmark Laboratory Experiments* is an example. The context is idealized by underlying parameters of variation in order to make inference easier. These methods attempt to address a space of possible situations rather than just particular situations that have been seen.

TABLE 11 Methods Used in Different Design Activities

Method	Problem	Artifact	Evaluation	Total
ANALYSIS				
Field Interviews	3		2	5
Benchmark Lab Experience			4	4
User Observations	1		2	3
Scenarios	3			3
Analytic Calculations			3	3
Work Flow Analysis	2			2
Objects and Actions		2		2
Comparative User Preferences			2	2
Formal Usability Targets		1		1
Comparative Field Experience			1	1
Questionnaires			1	1
Absolute User Preferences			1	1
SYNTHESIS				
Cut and Try		4	5	9
Use What You Build	3		4	7
Steal and Modify		4		4
Design for Yourself	3			3
Participatory Design	2			2
Generic Functions		2		2
Special Projects	1		1	2
Alpha Testing			2	2
Try to Break It			2	2
Reaction to Mockups	1			1
Beta Testing			1	1

A third dimension is the *Idealization of Representation.* At one end are methods that do not use a representational articulation of the design or the situation or the behavior. *Use What You Build* and *Field Interviews* are examples of these. At the other end are methods that have a notational representation,

FIGURE 5



Three dimensions of variation along which methods for developing human-computer interaction systems can be placed. A fourth dimension, not shown here, concerns the purpose of the method.

such as *Analytic Calculations* or *Objects and Actions*. In between are methods that have some sort of informal representation, as the use of *Scenarios* to represent a situation.

When we categorize all the methods of Table 11 by these three dimensions, we obtain Table 12. A few synthetic methods, such as *Steal and Modify*, are applied across contexts, and they have been listed in a new row, "Context-Free Methods" in the table. Otherwise, the methods of our inventory fit into the scheme. Figure 6 takes this analysis one step further and shows miniaturized versions of Table 12 for each of the papers in Table 2. A cell in the table is shaded in if the paper used at least one method in that cell. The analysis of the New TAO Workstation used only Analytic methods; the design of the Meetingware system used only Inarticulate Synthetic methods. Otherwise, the rest of the systems used an eclectic combination of different cells.

Essentially, these three dimensions of methods—Analytic vs. Synthetic, Idealization of Context, and Idealization of Representation—represent tradeoffs among three pragmatic variables: (1) power, (2) richness, and (3) cost. Methods oriented toward the external context and the methods with inarticulate representation have access to the full, richly textured experience of users and situations. But the analysis that can be done with these is limited. As the

TABLE 12 Dimensions of Variation for Methods Used

Idealization of Context	Idealization of Representation		
	INARTICULATE	INFORMAL	FORMAL
ANALYTIC METHODS			
Constructed		• *Benchmark Laboratory Experiments* • *Comparative User Preferences (ranking)* • *Absolute User Preferences (rating)* • *Questionnaires*	• *Analytic Calculations*
Self			
External	• *Field Interviews* • *User Observations*	• *Scenarios* • *Work Flow Analysis* • *Comparative Field Experiment*	
SYNTHETIC METHODS			
Constructed			
Self	• *Design for Yourself* • *Use What You Build* • *Alpha Testing* • *Try to Break It*	• *Design Before Coding* • *Comprehensive Design Description*	• *Formal Usability Targets* • *Objects and Actions Analysis* • *Generic Functions*
External	• *Participatory Design* • *Special Projects* • *User Reaction to Mockups* • *Beta Testing*	• *Usability Problem Tracking*	
Context-Free Methods	• *Steal and Modify* • *Cut and Try* • *UI Constraints First* • *Designer Management*		

FIGURE 6 Miniaturized versions of Table 12, showing which cells of the table are occupied by methods used in buildi this section.

TAO

Idealization of Context	Idealization of Representation		
	Inarticulate	Informal	Formal
ANALYTIC			
Constructed		●	●
Self			
External		●	
SYNTHETIC			
Constructed			
Self			
External			
Context-Free			

Rally

Idealization of Context	Idealization of Representation		
	Inarticulate	Informal	Formal
ANALYTIC			
Constructed		●	●
Self			
External	●		
SYNTHETIC			
Constructed			
Self			●
External			
Context-Free		●	

Meetingware

Idealization of Context	Idealizatior Representat
	Inarticulate
ANALYTIC	
Constructed	
Self	
External	
SYNTHETIC	
Constructed	●
Self	●
External	●
Context-Free	●

FreeStyle

Idealization of Context	Idealization of Representation		
	Inarticulate	Informal	Formal
ANALYTIC			
Constructed		●	
Self			
External	●	●	
SYNTHETIC			
Constructed			
Self	●		
External	●		
Context-Free	●		

Star

Idealization of Context	Idealization of Representation		
	Inarticulate	Informal	Formal
ANALYTIC			
Constructed		●	●
Self			
External	●	●	
SYNTHETIC			
Constructed			
Self	●	●	●
External	●	●	
Context-Free	●		

Total

Idealization of Context	Idealizatior Representat
	Inarticulate
ANALYTIC	
Constructed	
Self	
External	●●●
SYNTHETIC	
Constructed	●
Self	●●●
External	●●●
Context-Free	●●●●

representation of the situation moves toward more constructed context or toward more formal representations, they employ abstractions that selectively omit information (Restnikoff, 1989). These abstractions introduce real possibilities for error in understanding a situation. In return, they enable the use of more powerful operations on the representations. For example, in the Rally case, representing the design in terms of menu crossings enabled the team to construct a counting metric to rapidly eliminate many inferior designs. The alternative would have been direct inarticulate reactions from users experience of each in the field. In the case of Star, using the Keystroke Model allowed calculations to be done to predict expert performance at a time when no experts existed. Wisely, many projects resolve this tension by using combinations of methods situated along this tradeoff curve. This is the classic tradeoff in engineering and in science: *Analysis and deeper understanding depend on idealized representations that abstract away from the thing itself in all its particularity.*

The other tradeoff is cost. Methods vary enormously in cost. Fielding a full prototype with its customer relations and technical support team may be very expensive, for example, but also very revealing. Again, wisely, many projects deploy combinations of methods along the tradeoff curve. *Use What You Build,* for example, may be a cheap way to uncover many of the same flaws that would be discovered much more expensively by fielded prototypes. By using this cheap method, the fielded prototype can be used to uncover the more subtle problems.

The fourth dimension of methods concerns the things they are trying to accomplish. At any time, the design is in some state from problem identification through introduction (and beyond, actually). The methods are essentially techniques for determining that state or moving it to the next state. We have classified these by Problem Identification, Artifact Construction, and Evaluation. There is no accepted list that further breaks out the things to be accomplished in a design. Table 13, therefore, puts forward a proposal for a set of mileposts in each of these categories. Up to this point, the analysis has been confined to methods that were actually used to develop the systems in Table 2. The mileposts in the table reflect, in addition, experiences in systems research at Xerox PARC.[5] The first column in the table lists the milepost, the second column lists evidence (not the only possible) that the milepost has been reached. The last column in the table lists methods from this paper that either can help to reach the milepost or could help to tell if the milepost had been reached. Projects can, of course, skip some of the mileposts by assuming that attempted demonstration would be positive. This saves time, but adds risk. Thus, some projects are done intuitively and succeed, showing that the intuitions were correct about the state of the project. But other projects assume mileposts that later turn out not to be true. The classic example is for the

TABLE 13 State-of-the-Design Mileposts

Milepost	Evidence for	Applicable Methods
PROBLEM MILEPOSTS		
1. What is the work practice?	Naturalistic descriptions that explicate interrelationships and context	• *Field Interviews* • *User Observation* • *Participatory Design* • *Design for Yourself*
2. How can the essence of the activity be abstractly characterized?	Abstractions that characterize the essence of the activity. Identification of key factors and parameters of variation	• *Scenarios* • *Work Flow Analysis* • *User Reaction to Mockup* • *Objects and Actions*
3. What is the critical problem?	Identification of solvable problem that will lead to high payoff	
ARTIFACT CONSTRUCTION MILEPOSTS		
1. What is the basic invention idea?	Inspiration for invention, expressed notebook drawing, napkin, seminar, or hallway conversation	
2. What is the Application Internal Model?	Internal clockworks model of the application worked out	• *Objects and Actions* • *Steal and Modify* • *Comprehensive Design Description*
3. What is the Design User's Conceptual Model?	Model of the application as intended to be seen from the user	• *Objects and Actions* • *Steal and Modify* • *Comprehensive Design Description* • *Generic Functions*
4. Is the artifact feasible?	Wizard can use	• *Design for Yourself*
5. Can the grubby details be overcome?	Friends of Wizard can use	• *Cut and Try* • *Use What You Build*
6. Is the artifact robust?	People who don't even know the Wizard can use	• *Alpha Testing* • *Try to Break It*

Milepost	Evidence for	Applicable Methods
7. Has the artifact been modularized and packaged?	Documented API	
8. Is the technology use replicable?	Demonstration that same technique can reliably be applied again with the same good result	
EVALUATION MILEPOSTS		
1. Is the artifact usable?	Users can actually put to use (including opening box and setting up)	• *Benchmark Laboratory Experiments* • *Alpha Testing*
2. Is the artifact useful?	Evidence it has solved the problem. Voluntary regular use	• *Use What You Build*
3. Has the artifact solved a problem in a significant domain?	Propagation of system to larger groups of users	• *Field Interviews* • *User Reaction to Mockups*
4. Does the solution have commercial utility?	Sales	

designers to assume they understand the problem the design is to solve without actually taking the time to visit the field and understand the users.

The list in Table 13 is a tentative reorganization of the analysis of Table 12. The point it tries to make is that: *Methods are not ends in themselves, but means to certain ends.*

There are alternative means to accomplish those ends. The demand for a certain function, for example, could be examined analytically through interviews and observation, or it could be examined synthetically by trying to build experimental prototypes and iterating based on experience. Systems and user interfaces that are successful are not successful because they use a particular magic method, but because, using some collection of methods, they managed to accomplish certain ends, such as identifying the demand, making the system usable, and making the system robust. There are alternative means by which these ends could be accomplished, probably including methods associated with radically different philosophies. But there are not alternative ends to accomplish. This list of ends (and some practical ways of accomplishing them) is the real thing for which we are looking. Table 13 is a first attempt.

Pioneers and Settlers

A theme that comes through these analyses is the difference between Settler Systems seeking incremental advances in user interface technology by using known or incrementally improved techniques on new problems and those Pioneer Systems seeking larger increments in the state of the art. This division is one of degree and continuum since many systems combine a little Pioneering with a lot of Settling. Because of the difficulty of developing Pioneer Systems, many companies avoid investing in them. This is expressed in the quotation: *The Pioneers get the arrows, the Settlers get the land,* from Guy Kawasaki at the beginning of this paper. But many of the breakthroughs in human interfaces that have enabled the expansions of entire industries have come from such pioneering research. It is doubtful whether the new industrial research philosophies, such as "third-generation research management" or "research as part of development," which stress the tight coupling of research to product development, would have produced such Pioneering advances although they might have been more efficient at producing incremental Settler advances.

Figure 7 again shows Figure 1, only this time the SSI Invention rating has been indicated for each system and also the funding sponsorship. We have also added to the figure an indication of funding sources.

From the figure, we note the critical role of federal government sponsorship of the early Pioneer Systems. This relationship stands out quite distinctly when we plot SSI Invention rating as a function of development phase (Figure 8). On the figure, we have indicated funding source. High innovation occurs early. In essence, government, especially ARPA, funding laid the seeds of the graphical user interface revolution of the 1970s and 1980s. Department of Defense program managers, especially ARPA program managers, did actually *do something* about the problems and potential of human interfaces with computers. Government funding of advanced human-computer interaction technologies built the intellectual capital and trained the research teams for the Pioneer Systems that, over a period of 25 years, revolutionized how people interact with computers. Industrial research laboratories at the corporate level in Xerox, IBM, AT&T, and others played a strong role in developing this technology and bringing it into a form suitable for the commercial arena. The really big advances in human-computer interaction were worked out in these Pioneering Systems.

Most systems (as we saw from Table 3) are Settler Systems, and so these systems are an important consumer of methods, especially engineering methods, for the routine production of new designs. But the Pioneer Systems drive the technology. Pioneer Systems are more likely to be associated with methods for getting deep understanding of their task domains or to be associated

FIGURE 7 Each system in Figure 1 has been assigned an approximate magnitude on the invention scale and has been annotated with respect to funding source. Much of the earlier work with the highest impact was government funded. Work funded by development organizations tended to be lower on the SSI scale.

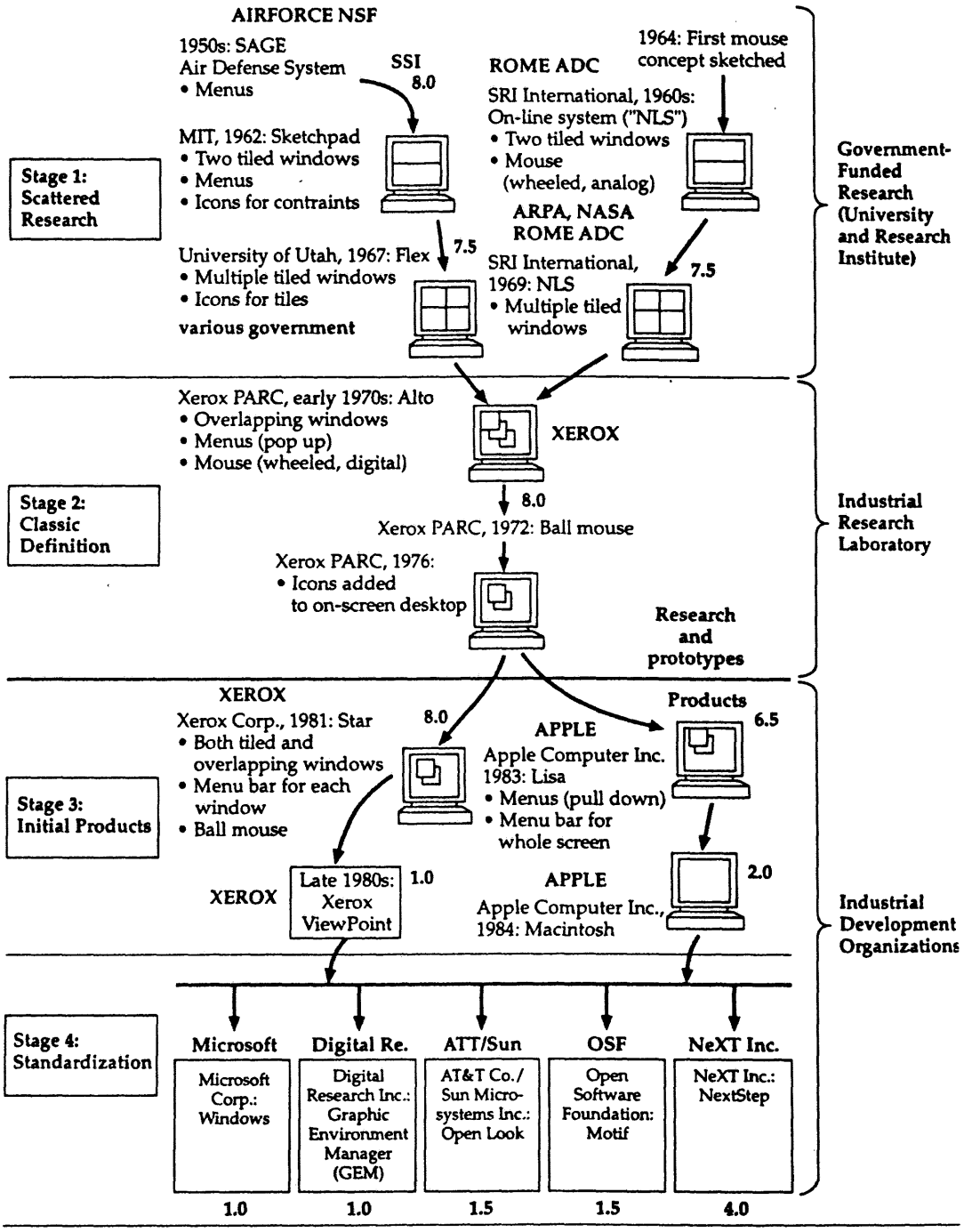
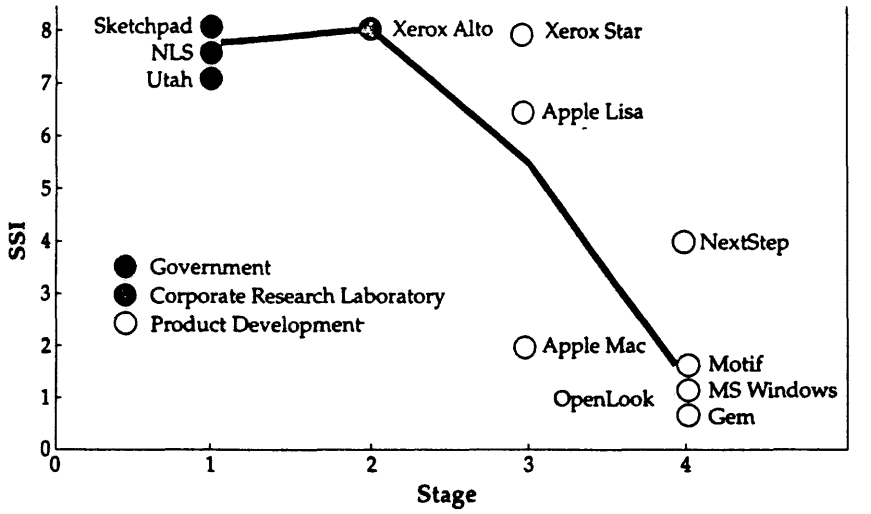


FIGURE 8



Ratings on the Seismic Scale of Invention (in Figure 7) as a function of user interface stage and funding source. Although based on subjective ratings, the basic pattern is clear: Pioneering invention for the cases examined tended to take place in federally funded research projects and central corporate research centers. *Based on data collected from the research community by Brad Myers and on listed acknowledgments.*

with new technology or deep understanding of some aspect of technology. Methods oriented toward developing the next Settler System are likely to be inadequate for some aspects of Pioneer Systems.

Conclusions

Now let us summarize our conclusions. We began by looking for methods used to design successful interactive computing systems. As a consequence, we needed to characterize what it means for such a system to be successful. Our analysis of success led us to distinguish three varieties: *invention success, engineering success,* and *innovation success.* In fact, we suggested measuring impact quite literally with a Seismic Scale of Invention. Systems that score high on this scale we called *Pioneer Systems;* those that score low we called *Settler Systems.*

Most user interfaces are Settler Systems. They seek engineering success: systems that meet their defined objectives for usability or human performance (and objectives that are reasonable). Most methods in the HCI literature are oriented to the production of engineered Settler Systems.

Innovation success, success in the arena of engagement between a system and its public, is the third kind of success we considered. Innovation success is an outcome of a much wider set of factors. An innovation success requires four basic ingredients to come together: technical merit, embodiment merit, operational merit, and market merit. Although invention success and engineering success are forces for innovation success, excellent methods used by excellent designers can nevertheless fail to produce innovation success because the failure can derive from business failures in the larger context, such as market position, distribution alliances, resources, or regulatory action. The converse can also be true.

The design of an interactive system can have any and perhaps several of invention, engineering, and innovation success, but methods that produce engineering success for Settler Systems, usability testing, for example, are not themselves sufficient for producing invention success for Pioneer Systems.

To illuminate the role of methods in successful system design, we noted that the process of invention flows through several phases. We simplified the phases down to three principal ones: problem identification, artifact building, and evaluation. We used these phases to organize the methods used to build the systems under analysis. Most systems used several methods. We noted particularly the wide use of the *Cut and Try* method for iterating designs and the *Steal and Modify* method for harvesting experience built up earlier in the food chain of ideas.

Beyond the roles they support, we suggested methods could be classified along four dimensions. First is whether the method is analytic or synthetic. Second is the degree of idealization of design context, and third is the degree of idealization in the design representation. These latter two dimensions reflect the tension between the richness of direct contact with a design situation, on the one hand, versus, on the other hand, the analytical power that arises by working with idealized abstractions. A related tension is between using better, but costlier, methods versus less good, but cheaper, ones. These tensions were addressed in the designs under examination by using combinations of methods differing in their tradeoffs. The fourth dimension of method classification is what the method is trying to accomplish. The desired accomplishments can be arranged as design mileposts. It is really these mileposts that are the goal of the methods. There are many alternative methods that can be used to achieve the same milepost, but there are not alternative mileposts.

Let us now come back to our original problem of identifying the methods that produced successful HCI systems. As we have seen, we cannot simply peer into systems that are successful in their markets or missions and hope to derive directly the methods that made them so—the original premise of this book. The critical seeds of success were often sown long ago in a Pioneer

predecessor system and may even, in some cases, form tacit or embedded art from which the current system was built. Only by examining the larger context of technological evolution and mission engagement can we see through to the lessons we seek to learn. In summary, the lessons might be stated:

If you are interested in just a single point design, exploit the food chain of ideas, and employ established methods to accomplish the design mileposts. Some mileposts may be assumed, but this always increases risk. Analyze the design in terms of the four types of merit underlying innovation success in order to identify vulnerabilities.

If you are a company or funding agency taking the long view, fund research in Pioneer Systems to build up the food chain of ideas.

At any rate, adopt a combination of methods in order to diversify risks, minimize cost, and accumulate as much insight into the design as practical.

A few caveats about the limitations of our analysis: The analysis in this paper has been confined to methods in the systems studied. Of necessity, a certain amount of interpretation is involved in extracting and assigning methods, but I believe that we are still able to use this intrinsically noisy data to discover the larger patterns of methods for successful systems. I believe these larger patterns would still hold, even in the face of some particular errors of interpretation for a particular system. Further, there are other methods in use that did not happen to occur visibly in these systems. I believe that these could largely be fitted into the patterns described, but it would be an interesting exercise to see if other patterns emerged as well.

We have discussed and rated HCI dialogue techniques as the basic inventions units of HCI. We could do the same for methods themselves although this would be far harder. There is a food chain of ideas in methods just as there is for systems; processes can be key innovations just as much as things. An analysis of this sort would be required to do full justice to the Atwood, Gray, and John paper on CPM-GOMS instead of focusing on the system for which the method was used, as we have had to do here.

Finally, a word about timing and how it affects success. Good engineering of Settler Systems is always relevant, but the timing of Pioneer Systems is more complex. Technologies proceed by punctuated evolution: They go through relatively short periods of rapid change followed by long periods of stability. The impact of Pioneering systems, therefore, depends, in part, on timing. At certain periods, new ideas (such as scroll bars on windows) slip easily into the new technology, but later superior ideas find it difficult to dislodge the incumbent. Methods selected for design must always be sensitive to their ecological context and tread skillfully between new species and established competitors.

Notes

1. John Whiteside presented this work at the workshop, but the paper is not included in this volume. See Whiteside (1993).

2. Smalltalk was not commercially available, so it could not have been a commercial success during the period referred to.

3. To get the fullest possible picture of the design process for FreeStyle, in addition to the paper in this section, I have drawn on Francik and Akagi (1989), Francik et al. (1991), and Levine and Ehrlich (1991).

4. To get the fullest possible picture of the design process for Star, in addition to the paper in this section, I have drawn on Bewley et al. (1983), Smith et al. (1982), Verplank, (1988), Johnson et al. (1989), a lengthy telephone interview with Charles Irby, the head of the design team, and my own recollections.

5. The Artifact Construction and Evaluation lists derive from discussions with, and are the inspiration of Rick Beach. They partially reflect PARC experience in commercial printer development. The scale items "Wizard can use," "Friends of Wizard can use," "People who don't even know the Wizard can use" are traditional PARC measures of research system maturity, invented by Ed McCreight.

References

Apple Computer (1987). *HyperCard User's Guide.* Cupertino, CA: Apple Computer.

Bewley, W. L., Roberts, T. L., Schroit, D., and Verplank, W. L. (1983). Human factors testing in the design of Xerox's 8010 Star office workstation. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 72–77.

Bolt, R. A. (1984). *The Human Interface.* Belmont, CA: Lifetime Learning Publications.

Burgleman, R. A. and Maidique, M. A. (1988). *Strategic Management and Technology of Innovation.* Homewood, IL: Irwin.

Card, S. K. (this volume). Pioneers and settlers: Methods used in successful user interface design.

Card, S. K., Mackinlay, J. D., and Robertson, G. G. (1991). A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems* 9 (April 2), 99–122.

Card, S. K., Moran, T. P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum.

Carr, R. and Shafer, D. (1991). *The Power of PenPoint.* Reading, MA: Addison-Wesley.

Cooper, G. E. and Harper, R. P., Jr. (1969). The use of pilot rating in the evaluation of aircraft handling qualities (AGARD-597). Paris, France: Advisory Group for Aerospace Research and Development. (DTIC No. AD689722).

Dahl, O.-J. and Nygaard, K. (1966). SIMULA—An Algol-based simulation language. *Communications of the ACM* 9(9), 671–678.

Engelbart, D. C. and English, W. K. (1968). A research center for augmenting human intellect. *AFIPS Proceedings of the Fall Joint Computer Conference* (Vol. 330), 395–410.

Francik, E. and Akagi, K. (1989). Designing a computer pencil and tablet for handwriting. In *Proceedings of the 33rd Human Factors Society Annual Meeting* (Denver, October 16–20). Santa Monica, CA: The Human Factors Society, 445–449.

Francik, E., Rudman, S. E., Cooper, D., and Levine, S. (1991). Putting innovation to work: Adoption strategies for multimedia communication systems. *Communications of the ACM* 34(12) (December), 52–63.

Fisher, S. (1989). The AMES Virtual Environment Workstation (VIEW). *SIGGRAPH '89*. Course #29 Notes.

Furness, T. A. (1986). The super cockpit and its human factors challenges. *Proceedings of the Human Factors Society, 30th Annual Meeting*. Santa Monica, CA: The Human Factors Society, 48–52.

Gould, J. D., Boies, S. J., Levy, S., Richards, J. T., and Schoonard, J. (1987). The 1984 Olympic Message System—A test of behavioral principles of system design. *Communications of the ACM* 30(9) (September), 758–769.

Herot, C. F. (1980). Spatial management of data. *ACM Transactions on Database Systems* 5 (December 4), 493–514.

Hughes, T. P. (1983). *Networks of Power, Electrification in Western Society, 1880–1930*. Baltimore, MD: The Johns Hopkins University Press.

Johnson, J., Roberts, T. L., Verplank, W., Smith, D. C., Irby, C., Beard, M., and Mackey, K. (1989). The Xerox Star: A retrospective. *IEEE Computer* 22 (September 9), 11–26.

Kaare, C. (1983). *The Unix Operating System*. New York: John Wiley & Sons.

Kay, A. C. (1977). Microelectronics and the personal computer. *Scientific American* 237(3) (September).

Kay, A. C. and Goldberg, A. (1977). Personal dynamic media. *IEEE Computer* 10 (March 3).

Lampson, B. W., ed. (1976). *Alto User's Handbook*. Palo Alto, CA: Xerox Palo Alto Research Center.

Lampson, B. W. (1988). Personal distributed computing: The Alto and Ethernet software. In A. Goldberg (ed.), *A History of Personal Workstations*. New York: ACM Press.

Levine, S. R. and Ehrlich, S. F. (1991). The FreeStyle system: A design perspective. In A. Klinger (ed.), *Human-Machine Interactive Systems*. New York: Plenum Press.

Miller, R., Miller, R., and Lovick, S. (1989). Cosmic Osmo. Computer program for Macintosh. Cyan Productions and Activision.

Newell, A. (1992). Unified theories of cognition and the role of Soar. In J. A. Michon and A. Akyurek (eds.), *Soar: A Cognitive Architecture in Perspective*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Newman, W. and Sproull, R. (1973). *Principles of Interactive Computer Graphics*. New York: McGraw-Hill.

Perry, T. S. and Voelcker, J. (1989). Of mice and menus: Designing the user-friendly interface. *IEEE Spectrum* (September) 46–51.

Restnikoff, H. L. (1989). *The Illusion of Reality*. New York: Springer-Verlag.

Robertson, G., Newell, A., and Ramakrishna, K. (1981). The ZOG approach to man-machine communication. *International Journal of Man-Machine Studies* 14(4) (May), 461–488.

Scheifler, R. W., Gettys, J., and Newman, R. (1988). *X Window System*. Bedford, MA: Digital Press.

SIGGRAPH (1990). Computer Graphics Achievement Award: Richard Shoup and Alvey Ray Smith. *Computer Graphics* 24 (August 4), 17–18.

Smith, D. C. (1977). *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*. Basel: Birkhäuser Verlag.

Smith, D. C., Irby, C. H., Kimball, R. B., Verplank, W. H., and Harselm, E. F. (1982). Designing the Star user interface. *Byte* 7(4), 242–282.

Stallman, R. (1987). *GNU Emacs Manual, Sixth Edition, Version 18*. Cambridge, MA: Free Software Foundation.

Sutherland, I. E. (1963). *Sketchpad: A Man-Machine Graphical Communication System*. Ph.D. thesis, MIT. Reprinted by Garland Publishing, Inc., New York, 1980.

Tesler, Larry (1981). The Smalltalk environment. *Byte* (August).

Verplank, W. (1988). Designing graphical user interfaces. Tutorial notes, *ACM CHI '88 Conference on Human Factors of Computing Systems*, Washington, D.C.

White, G. R. and Graham, M. B. W. (1978). How to spot a technological winner. *Harvard Business Review* (March–April), 146–152.

Whiteside, John. (1993). The phoenix agenda: Power to transform your workplace. Essex Junction, VT: Omneo, xvi, 318p.: ill.; 24 cm.

Williams, G. (1983). The Lisa computer system. *Byte* 8 (February 2), 33–50.

Human-Computer Interface Design

Success Stories, Emerging Methods, and Real-World Context

Edited by

Marianne Rudisill
NASA Langley Research Center

Clayton Lewis
University of Colorado

Peter G. Polson
University of Colorado

Timothy D. McKay
Gateway 2000

Morgan Kaufmann Publishers, Inc.
San Francisco, California

Sponsoring Editor Michael B. Morgan
Production Manager Yonie Overton
Production Editor Cheri Palmer
Editorial Coordinator Marilyn Uffner Allen
Production Books By Design, Inc.
Cover Design Ross Carron, Carron Design
Text Design Books By Design, Inc.
Compositor Sue Cologgi
Copyeditor Robert Fiske
Proofreader Megan McDowell
Indexer Marilyn Rowland
Printer Courier Corporation

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA
Telephone 415 / 392-2665
Facsimile 415 / 982-2665
Internet mkp@mkp.com
Web site http://mkp.com

Printed in the United States of America

00 99 98 97 96 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Human-computer interface design : success stories, emerging methods, and real-world context / edited by Marianne Rudisill...[et al.].
p. cm.
Report of a workshop held July 1991 in Boulder, Colo.
Includes bibliographical references and index.
ISBN 1-55860-310-7
1. Human-computer interaction. 2. User interfaces (Computer systems) I. Rudisill, Marianne.
QA76.9.H85H8653 1995
005.1'2—dc20 95-20362
CIP