Usability as a pawn in the power politics of a venture firm.

Product, Process, and Profit: The Politics of Usability in a Software Venture

Barbara Mirel
National Center for Patient Safety
Veterans Health Administration (10X)
2215 Fuller Road
Ann Arbor, Michigan 48105
barbara.mirel@med.va.gov

Abstract

In research and in practice, usability specialists commonly target the technology—user-interfaces and help—as the main arena for bringing about usability improvements. However, usability improvements depend on more than innovative and user-centered technical designs and implementations. Equally important for creating useful and usable software are the social and political forces that shape the development context. These forces give rise to leadership conflicts, factional disputes, renegade efforts, alliances and betrayals, all of which profoundly influence whether usability improvements will be supported and sustained within and across projects. This essay presents and analyzes a case history of a software start-up company in which usability achieved a Pyrrhic victory, triumphing only in the short run because of social and political forces.

K.4.3 *Organizational impacts*
Keywords*: usability, political support, goal conflict, sociology*

Introduction

This is a story of clashing cultures, of struggles between a handful of start-up, coding cowboys and a group of talented, steady-eddie developers who joined together in a venture to produce pioneering software. This clash of cultures spurred breakthrough gains in usability but it left casualties in its wake, casualties grave enough to arrest continued usability innovation. This story recounts a moment of triumph and glory for usability innovations that faded because of organizational processes and power relations.

I narrate this story as the human factors lead in this start up venture. I address one basic question: What does it take technically *and* organizationally to create breakthrough innovations in usability for computer-supported, complex tasks? By technically, I mean designs and implementations in the software product itself—its features, functions, architecture, interfaces, and help. By organizationally, I mean the complex network of social and political relationships, structures, processes, policies, norms, values and rules that shape choices about software design and implementation (Giddens, 1984). As I hope this story makes clear, changes technically and organizationally are equally necessary to bring about usability innovations. Technical and organizational systems are intricately intertwined (Latour,1988; Thomas,1994; Knights and Murray, 1994; Feenberg, 1995; Jackson, 1996; Frost and Egri, 1995; Star, 1995). The dynamics of change and resistance that flow within and between them make or break advances in usability. If, as is often the case in the usability literature, we pay attention only to innovating the technology, we will get an incomplete and misleading picture of what it takes to create useful software (Nielsen, 2000; Spool et al., 1999; Rubin, 1994).

Temptations loom large for us to devote our energies almost entirely to technical redesigns and enhancements especially when the need for these efforts is so apparent—as it is in the case of building software that is useful for users' complex tasks, end-to-end. These cases, however, are precisely the ones that most need social and political changes to accompany technical efforts. In these cases, technical efforts for imposing usability require contextual and user-centered approaches that rattle many developers' core beliefs, methodologies, and claims to control and turf (Beyer and Holtzblatt, 1998; Nardi, 1996; Anderson, 1994; Ramey et al., 1996; Rheinfrank et al., 1991; Brown and Duguid, 1991; Denning and Dargin, 1996; Nyce and Lowgren, 1995). If adequate organizational changes do not accompany these approaches, resistance to them may very well stifle or derail even the most necessary of usability innovations.

Therefore, to build usefulness and ease of use into software from the start, social and political innovations are equally if not more important than technology innovations. I say more important because, as my story shows, in the dynamic interplay between technical and organizational systems, the influence that one system has on the other is asymmetrical. Social structures and political processes have far more power in shaping the technical possibilities that are left open or foreclosed than technology innovations have in directing or re-engineering organizational work arrangements, power relations, and policy. Given this unequal influence, the experiences I relate underscore that usability experts must deliberately strive as much for political innovations as technological ones.

Usefulness as a Technology Breakthrough

Before recounting this story, I need to set the stage briefly with two explanations. The first focuses on technological innovations for usability.

When, in this story, I talk about breakthrough innovations in usability, I mean innovations in usefulness for complex tasks—for the nonroutine, problem-solving work of everyday, not early-adopter, users (Holland, 1998). Usefulness is probably the hardest but arguably most important factor to attain in usability. It involves representing in an application users' own models of their work-in-context, their approaches to it, and their identities in it (Agre, 1997; Coyne, 1995; Johnson, 1998). A mismatch between the task models built into a program and users' actual ways of thinking about and doing their work is perhaps the least redressed of all usability problems (Liddle,1996; Cooper, 1999; Norman, 1999). Improvements in this area, therefore, can provide dramatic breakthroughs.

Improvements for usefulness, however, are often nontrivial to implement. It takes intensive effort, for instance, to communicate situated tasks rather than operation-level functions; to transform context-free interface objects into "organizationally relevant things;" to provide domain-specific content, intelligent assistance, and screen cues for real world choices and purposes; and to structure places not just spaces for electronic work (Coyne, 1995; Johnson-Eilola, 1997; Albers, 1999; Agre, 1997; Ramey et al., 1996, Johnson et al., 1995). These designs can be demanding and seem to leave no time for realizing complementary changes in the organizational system. Yet, as my second explanation now discusses, without these organizational changes, attempts to achieve breakthroughs in usefulness may come to naught.

breakthrough innovations were achieved but the unintended losses accompanying these innovations were too deep and corresponding organizational changes too shallow to sustain the usability visions and gains.

Enter Organizational Politics

Breakthrough innovations in usefulness are unavoidably political. By definition, they challenge the status quo in development organizations. Tacitly or overtly, proponents of software innovation charge that historical precedents, institutionalized norms, and structural regularities are obstructing success. Organizational routines are questioned, previous versions of the technology get disparaged, various groups' stakes in existing and new approaches are exposed, and previously uncontested organizational objectives and processes associated with the technology at hand get scrutinized.

These tensions are evident in three types of decisions about technology, decisions shaped by prevailing interests and worldviews (Thomas, 1994). They determine whether a usability innovation will occur, and, if so, what trajectory it will take. They include:

1. Decisions about the defining architecture of the product.
2. Within that architecture, decisions about scope, requirements, and features.
3. Decisions about how to implement targeted features and functions.

The interests that dominate and control the outcomes of early decisions about architecture and features ultimately wield the greatest influence over the trajectory of innovation (Latour, 1988; Thomas, 1994). In my story, decisions about implementation were far less consequential, even though such implementation issues as selecting and successfully laying out effective user interface (UI) controls or naming them well are commonly cited in the literature as *the* arena for determining the fate of usability (Czerwinski et al, 1999; James, 1999; Forrester, 1998; Wiklund, 1994). In my case, implementation decisions occurred well after most of the possibilities for usefulness were already carved out and constrained.

My story focuses on the processes more than the product of innovation—processes that were frequently unpredictable and that led to unintended consequences and precarious usability gains. This case history is more than the story of one innovation project. I trace cumulative decisions across several projects to show their actual as well as forsaken outcomes, the possibilities that they opened as well as closed. This story culminates in a capstone project in which usefulness innovations occurred. But if I only told the project-based story of this innovation, the lessons it might suggest would be misleading. From an historical perspective, the project brought about usability gains and losses, and understanding these gains and losses is, in the long run, more beneficial to usability efforts than prescriptions for success. In this case history, breakthrough innovations were achieved but the unintended losses accompanying these innovations were too deep and corresponding organizational changes too shallow to sustain the usability visions and gains.

In telling this story, I've made all names fictitious. I've also included a timeline summarizing events in Appendix A.

The Quest

In December, 1997, Visible Solutions became a venture within Pyrrhtel, a large multinational telecommunications corporation. The goal of this venture was to become profitable enough in three years to spin off as a separate company by turning the interactive data visualizations developed through almost a decade of Pyrrhtel research into a commercial product—data analysis software for business users. The visualizations consisted of UNIX-based graphics that could display hundreds of thousands of records and that could be manipulated and linked by users to drill down for the purpose of discovering trends, outliers, and other relationships. In Pyrrhtel's R&D lab, interactive visualizations were an "experimental playground" for exploring huge sets of data for telecommunications problems such as visualizing call records to detect incidents of telephone fraud. Early in 1997, Pyrrhtel's upper level management saw great potential for channeling this "graphic playground" into a profitable "graphic theme park."

Before the venture was officially launched, a pilot group worked for several months on assorted commercial projects for business users such as visualizing software code in order to detect imminent Y2K problems. Three veteran upper level Pyrrhtel managers led this group, and they became the founding fathers of the venture. One of them, Stan, was my immediate boss, the venture's Vice President of Development.

To this triumvirate's credit, they realized that the home-grown Pyrrhtel group—including themselves—were socialized in an R&D culture and inexperienced in getting a software product to market quickly and competitively. Turning the powerful but opaque visualization technology into an accessible, easy to use product required the mindset and energies of a start-up company. These founding fathers, therefore, went outside of Pyrrhtel and hired a CEO who started with Visual Solutions in December 1997. One week later the venture officially launched.

Jack, the CEO, was a hard driving Harvard Business School grad who prided himself on knowing how to situate high tech in the market. At 35, he had already set up a number of small ventures, made them visible to investors quickly, and sold them for handsome profits. He was well-studied in talking about all the ingredients of a successful company—teamwork, collaboration, and openness.

My start date coincided exactly with Jack's, reflecting the two inseparable charges of the venture—to bring a product to market and to assure that this product would be accepted by users. Turning out a usable product was a top priority because, in the preventure days, the pilot group had gotten burned by unusable products. Most of their customers had rejected and returned the visualizations for lack of usability. A full time human factors position was created to address this problem, and I was hired.

In January, 1998, right after the birth of the venture, Jack began holding regular quarterly company meetings. At each meeting, without exception, he ritualistically told the story of who we were and where we came from. In the same words and with the same slides, he recounted our mission and our beginnings. Our mission was to build visual analysis software that would "revolutionize business decision-making" by helping everyday business analysts graphically interact with and interpret data that was otherwise too unwieldy to analyze.

In the beginning, Jack would declare, we had a really cool technology but not a usable product. Teams were formed, and we brought visualizations out of UNIX land and into the competitive world of Windows. We targeted two vertical—domain-specific—markets. We were becoming a market leader in each and would soon take the whole world of data analysis by storm. We were talking to real users, learning real users' needs and building for them. We were leaving behind a stodgy R&D mentality and becoming a really cool start-up.

When hardships occurred, Jack integrated the mishaps into the tale without missing a beat. Within fifteen months, both vertical products were abandoned. These abandoned projects, the CEO spun, demonstrated our flexibility. We decided to shift product direction but it was for the better, and we were adept at hitting the ground running. Most importantly, we were learning. We learned enough to improve the core graphics and the surrounding platform of interface controls to make them a marketable OEM product for developer users in other software firms. These developers would build our visualizations into their software and give our powerful interactive graphics widespread visibility.

Jack integrated our emerging experiences into his tale each quarter. When we dropped vertical products for niche markets, we moved to develop a horizontal—that is, generic problem-solving—product, an all-purpose data analysis tool for any business domain. It was as if we had found the holy grail.

In mythic fashion, disappointments, conflicts, and mistakes became the birth-pangs of conquering new and uncertain worlds. This story was a quest tale; its hero was the venture. This ritually told quest recounted the origin, purpose and journeys of the group. We went through trials and battled demons, all for the mission of bringing the boon of visualizations to society. In such a quest, conflicts and losses are inescapable.

But the problem with this quest tale—stirring as it was—was that conflicts really didn't get resolved. Nor did they get synthesized into some greater endeavor. Rather conflicts simply disappeared, and losses never had lasting effects. Mostly we just grew wiser and headed off in a better direction. It was a story of yea-saying. As in any ritual reciting, it was a re-enactment, aimed at renewing listeners' unquestioning assent at each hearing.

Beneath the surface, what was going on in this ritual recounting was the CEO upholding his version of events as inviolate. Unfortunately, reality belied his account. Frictions in the actual venture were consequential. They were also complicated and two-edged. They incited abuses of power, *and* they sparked creativity and innovation. Creatively, provocative debates and frictions inspired usability innovations as nothing else could. Were frictions, as in Jack's story, truly to disappear, that disappearance would have taken with it much of the vitality, vision, and sense of mission of the group. In fact, that is exactly what happened.

Factions Emerge

Contending factions were born with the venture. The first factional split emerged between January and April, 1998, when Jack started his tenure by hiring and firing a number of people. In these four months, he brought on board eight new employees, all of whom had worked for him at his last start-up company. These new hires quickly became dubbed "cronies," a term of resentment when it came from the mouths of some veteran Pyrrhtel people, a badge of honor when voiced by Jack and the new hires themselves. One reason why some Pyrrhtel people resented the cronies was that, in April, at the same time as Jack was hiring his former employees, he unexpectedly fired six Pyrrhtel people. His reason, he said, was a "lack of fit."

The firing prompted Stan, the Vice President of Development, to call an emergency meeting of the twenty developers in the venture—all of whom, except for two new cronies and me, were Pyrrhtel veterans. The meeting was aimed at quelling our fears that, in significant ways, "old Pyrrhtel" was different from new "start-up" and perhaps at risk. As the quest tale conveys, "start up" was the favored culture and inseparable from harmony and growth.

However, developers' fears were real as well as mythic. The new hires, perceived as a "start up" faction, immediately assumed positions of authority, mostly as directors or team leaders. Jack continued to hire cronies as the year wore on. The highest ranking of them, Kevin, became Vice President of Product Management. By the end of 1999, Kevin controlled the vision of both the venture and the product, ultimately vying for power with Stan, the Vice President of Development and founding father.

Stereotypically, "start up" stood for fastpaced development, speed-to-market, small and nimble teams, and an aversion to formal process. "Old Pyrrhtel," by contrast, revered process. It also stood for open debates among wide-ranging ideas, protracted decision-making, and a staunch commitment to diversity in the workforce.

The two factions came into conflict early on in regard to one of the first decisions about the visualization technology—how to define the scope, structure, and design of the Windows-based graphics.

Initial Tensions About the Scope and Design of the Core Graphics

This conflict pitted the top "old Pyrrhtel" developer, Rick, against a newly hired "start up" developer, Ben. One of the first cronies hired, Ben came aboard in January, 1998, as a much needed Windows and graphic user interface (GUI) expert. He immediately joined the already started project of converting the researchers' UNIX-based graphics into Active-X components.

Ben came to the team with his own ideas about structure and design. But Rick, the lead of the team, who was the star developer in the venture, had already instituted his own standards and design. They were based on the assumption that primary users of the graphics were technical developers—like himself and the researchers before him—and that these users wanted screen real estate devoted exclusively to data, with few if any verbal cues for interaction and guidance.

Ben's sense of Windows design and his experiences in creating commercial products ran counter to Rick's. Unlike "old Pyrrhtel" Rick who had never brought a commercial Windows product to market, "start up" Ben was experienced in developing commercial successes. However, he did not yet have the social or political clout to succeed against Rick.

To the core, these two programmers disagreed about whether primary users were to be OEM developers or business analysts and whether the library for storing the Active-X controls should be modular or interconnected. Technically, the implications of Rick's current design and structure were that they closed some possibilities for building enhancements for usefulness for business users later on—for instance, for giving them task-based entry points and a way to filter data before visually querying it.

These disagreements were political as much as technical. Rick, the team lead, had no intentions of willingly giving up his control over the library and graphics design. Ben, too, was politically motivated. He was the most expert developer in the venture in regard to usability and designing for end-users. If end-users were to be the primary target, he was sure to take the lead in development efforts. Embedded in the debate were distinct social and political interests, equal and competing talents, and contending computing ideologies but at this point they never became overt.

The controversy ended almost as soon as it began. Rick pulled rank and said that the design and library were not open for discussion. The criteria behind his selection of design and structure were not to be questioned—they simply were part of Pyrrhtel lore and technical wisdom, the legacy of talented researchers. Moreover, Rick noted, the venture's only immediate potential customers were developer users who would embed the visualizations in their own company's business systems. He was designing for them. Most developers in the venture shared Rick's sentiments and deferred to his status, skills and knowledge.

Despite this outcome, this disagreement only seemed to disappear. It resurfaced many times, with greater force each time. Tensions between these two developers continued for two years. Ultimately, "start up" Ben would displace" old Pyrrhtel" Rick in status, control, and privilege.

In this case, however, debates were checked almost before they began. The issue of who the primary users were—who to design for—instigated debate again two months later in March, 1998. This time the decision was about product architecture, and it spawned new factions and arguments.

New Factions Form Defining the Platform Architecture

To house the Active-X graphics and provide mechanisms for accessing and manipulating them easily, a platform needed to be built and its architecture defined. In March, 1998, Stan, the Vice President of Development, assigned another newly hired crony developer, Pat, to head the platform initiative, deliberately wanting new blood and fresh approaches at the helm. The organizational chart now looked as follows:

Unlike Rick, the "old Pyrrhtel" lead of the graphics conversion team, Pat as platform lead overtly considered designing primarily for business analysts instead of developers and held a series of meetings for all developers and product managers to debate this issue. By May, after two months of meetings on this issue, the majority of people agreed that the platform developers should build generic interaction features that would address what the group assumed to be overlapping needs of end-users and developer users. Most of the group was in full accord when Pat, the platform lead defined

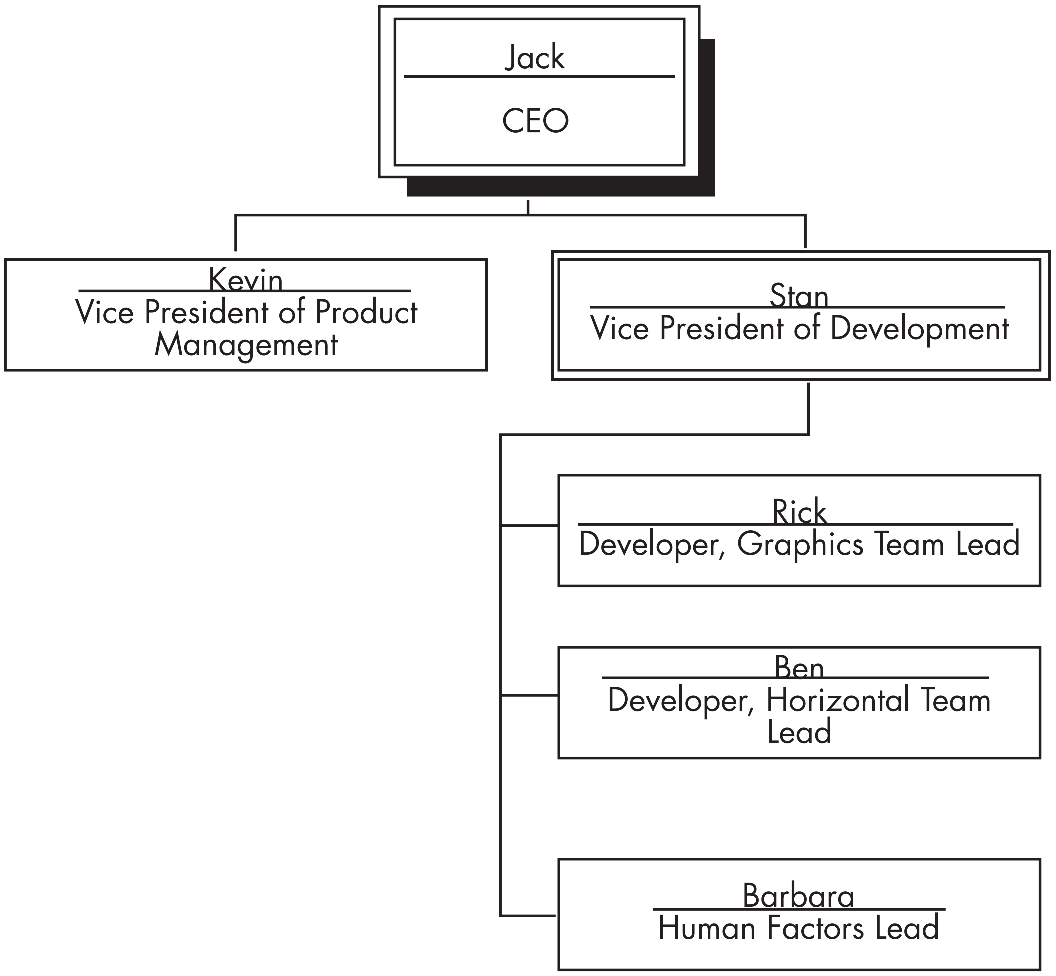
Jack
CEO
Kevin
Vice President of Product Management
Stan
Vice President of Development
Rick
Developer, Graphics Team Lead
Ben
Developer, Horizontal Team Lead
Barbara
Human Factors Lead

Figure 1. Organizational Chart of Main Players.

the platform as equally aimed at the two user groups—a two-in-one technology.

The defining aspect of this two-in-one architecture was the specification that each of its features and functions had to be scriptable. Scriptability would assure that end-users got their required functionality while developers were able to code as needed. In truth, the platform, almost from the start, became as biased toward developer users as the Active-X graphics were. Once Pat started building platform features, he gave priority to infrastructure issues that were vital to developers but not to end-users; he also tabled features that were too hard to script even if they were critical for supporting end-users. Bookmarking, for example, was nontrivial to script, and it remained tabled for sixteen months, from May, 1998, to September, 1999, even though at least four user studies between these dates requested it as a core end-user need for managing inquiry.

This definition of the platform as a two-in-one technology provoked some dissent. At the time, dissenters were a loosely knit group, cutting across "old Pyrrhtel" and "start up" lines. This group included me; Ben, the crony developer on the graphics conversion team; an "old Pyrrhtel" project manager for the vertical application team that I also was on; Kevin, the "start up" Vice President of Product Management and Stan, the "old Pyrrhtel" Vice President of Development.

Conceptually, we shared a bias toward what I'll call a "user first" over a "program first" perspective. The "program first" perspective to which we objected rested on the adage "if you *can* build a neat new feature, you *should* build it; users will welcome and adapt to it." The goal from a "program first" perspective was to build new and ever better features. Our "user first" frame of mind, by contrast, started with users' practices and purposes rather than with potential features. So long as basic features for user and system performance were built into the software, priority should be given to usability for task purposes. Since developers' task purposes—coding—and business users' purposes—problem-solving—differed dramatically, we all argued that one technology for both would have limited value for either. What was good for the goose was not necessarily good for the gander because, empirically, they were very different birds.

Kevin, the Vice President of Product Management, was in this "user first" camp primarily for market reasons, reasons shared by the rest of us, as well. From a market point of view, we wanted the venture to work on products that pushed past targeting OEM users. The OEM market was fine for short-term sales but the end-user market was far more profitable. If we were to succeed in this market, we needed to bias our products toward end-users not developers.

Over the months, this "user first" position crystallized. It argued for giving higher priority to usability improvements than to new, beguiling features. The position also stressed that nine out of ten times developers' designs do not capture users' needs. Finally, "user first" members held that prerequisites for success were to identify through usability studies a problem-solving content and to build it into end-user products. The "user first" group hoped to realize the mission of the venture by putting users at the center of every development effort. "Program first" people, by contrast, assumed that the dazzling power of the technology itself would win the market.

In some ways, platform-related oppositions between "user first" and "program first" echoed dissensions between "start up" and "old Pyrrhtel." "User first" and "start up," for example, both sought a robust architecture to support speedy development of end-user applications within the venture. But the two factions differed in how to design for this end. "Start up" stood for "thinking outside of the box" but user-centered design methodologies were too outside the box even for some of the most adventurous "start up" people. Pat, the platform lead, for example, thrived on inventive programming, but stopped short at interaction design. For most programmers, this leap was huge. For one it wasn't—Ben, the "start up" crony on the graphics conversion team. He became one of my two biggest allies in the user-centered camp and *the* pivotal player in effecting breakthroughs for usefulness a year later. My other biggest ally and usability champion was Stan, my boss. A hybrid of "user first" and "old Pyrrhtel," Stan valued in equal measure user-centeredness and an adherence to process to ensure quality products. This mixture would turn the tides against him in Fall, 1999.

At the time of the platform definition meetings, the objections of our minority "user first" group did not change the two-in-one platform definition devised by the "program first" group. But our objections did qualify it somewhat. Everyone agreed that at some point the two user groups' needs would probably stop overlapping, and, at that time, development would have to split, probably into two distinct platforms.

With this emerging "user first" faction and the two Vice Presidents' endorsements, usability was poised to take a more central role in product definition, design, and development.

A Crisis in Unilateral Decision-Making

Halfway into 1998, the venture settled into four product teams: one for the graphics (the same team that previously converted the graphics to Active-X), one for the platform, and two for vertical applications targeted to different end-users. Until July, 1998, Rick, the lead of the graphics team decided unilaterally what these four teams would build. Rick assigned selected features to the platform, and then Pat, the crony platform lead, similarly decided on his own what the priorities would be for the platform.

Both team leads' choices were biased toward developer users' needs. The assumption in the development division was that the graphics and platform teams with their six or so programmers apiece would build for generic interactions—the overlap between developer and end-users. End-users had domain-specific needs, as well, and the two or three programmers on each vertical team would develop for these needs.

In July, 1998, the vertical application teams completed their user and system requirements and submitted far more requests for generic building to Rick—the feature gatekeeper—than he had anticipated. These requests were aimed at supporting end-users' generic processes of inquiry, processes that cut across domains, for instance, managing inquiry, running and comparing parallel streams of inquiry, going off on a tangent and coming back, interrupting work and returning hours later, and socially negotiating the meaning of displays. These end-user requests were hard to script, and they competed with other technically difficult features required for developer users such as thin client architecture. Rick, the graphics team lead, tried to throw the end-user requests back over the wall to the vertical teams.

The vertical teams protested vociferously. From them came a burgeoning "user first" challenge to the graphics and platform teams leads' bias toward developer users and control over priorities. In addition, a "start up" line of attack was mobilizing against what appeared to be Rick's "old Pyrrhtel" tactics of adhering to unquestionable processes for making decisions and setting criteria.

In this July, 1998, crisis, two months after the decision to define the platform as a two-in-one technology, several all too familiar questions still begged for answers. For example, what did the two-in-one scope mean? Now that end-users' requirements were so numerous, was it still a feasible scope? What team should build for generic problem-solving processes? How were priorities to be distributed between developer users and end-users, who decided, and based on what criteria?

Had these questions finally been addressed head on, work groups might have been reorganized to put more programming resources on the vertical teams and authority for decisions might have been redistributed, both leading to a shift in bias and priority toward end-users. With ample resources and authority, designs for usefulness might have been built into the vertical applications from their inception as success demands. Yet, despite the opportune time, these fundamental questions did not become the focus in resolving this conflict. Rick, the graphics team lead, made a strategic move to appease objectors while still maintaining his control. He succeeded masterfully, and the cause of usefulness was set back considerably.

The Chits Hit the Fan

In addressing the unrest in July, 1998, Rick avoided substantive issues and focused instead on the surface problem of unilateral decision making. In what seemed to be a strategy of placating the "start up" group by appearing to be open to new processes, Rick proposed an allegedly democratic system for deciding priorities—a "free market" system of chits. Each team would get a budget of chits based on the projected revenues for their respective products. Teams would use these chits to buy features that they wanted and needed most. The price of the features, calculated by Rick as head developer, was to equal the effort it would take to build them. Two teams could pool their chits for the same feature and have more left over for other requests. In this way, teams would set their own priorities. The budgets for the graphics and platform teams combined were 168 chits; the vertical teams collectively had 42.

The development group was taken in by the game quality of this proposal. Real conflicts got deflected into a niggling over rules. Little was said about the system's viability or credibility, about the teams' woefully unequal budgets or the assumptions on which they were based.

It was a classic example of emperor's new clothes. No one stated straight out (a) that Rick and Pat, the graphics and platform team leads, were strongly biased toward developer users and had a "program first" mentality; (b) that these biases were favored 4 to 1 in the proposed budgets; and (c) that Rick was doing all the calculations. A couple of people did protest, myself included, but Stan, the Vice President of Development, dismissed our objections. In private conversation with me he indicated that he cared little about whether the group used a chit or some other system. What he did care about was mending the persistent "old Pyrrhtel" and "start up" divisions, especially the mounting tension between Rick, the graphics team lead, and Ben, the graphics team crony. The VP of Development believed the chit system would help to bridge the "old Pyrrhtel"/"start up" split because Rick was making a grand gesture to change his "old Pyrrhtel" ways.

The inequities of this "participatory system" became evident five months later. In December, 1998, after end-user requirements predictably got short shrift in the purchased features, the first vertical application team collapsed for want of interested users. Weeks later, the other vertical team's—my team's—trial version with alpha users evoked enough complaints about its lack of usefulness to send us back to the design table for another month. From our alpha user tests and observations, we now knew better than any team yet what end-users needed and wanted for visual analysis in context. Our new design consequently resulted in even more generic problem-solving requirements but without a chit budget to buy them.

To try to close the chasm between what users expected and what the core visualizations provided, our vertical team moved into the new year and spent all of January, 1999, building usability improvements into our product with whatever resources we could muster. Three events around us, however, converged to shut down this project for good in February. First, a partnership essential for getting the vertical application to market went sour, partly because the visualizations were too sparse in usefulness. The executive team decided against mending this partnership.

Their vote of no confidence was tied to the month's second event—the beginning efforts of Jack, the CEO, to woo venture capitalists a surprising year and a half ahead of schedule. Jack deemed our vertical application too shaky to impress prospective investors. He claimed that a graphics-and-platform toolkit for OEM users was a better bet because it would entice investors.

Simultaneously, Product Management could conceive a different, sexier end-user product to ultimately draw them in.

This strategy led to the third event of the month. At the end of January, Product Management issued a forty-five page product plan and feature list, giving clear priority to an OEM product for general availability (GA) release in May. It was a bold play by Kevin, the Vice President of Product Management, to claim decisions about features and priorities for himself. A week after the plan was circulated, Stan, the Vice President of Development, officially disbanded our vertical team. Ironically, the redesigning that we did in January won a software design award from a trade magazine a year later.

The Imperfect Storm

At the end of January, 1999, three successive days of meetings focused on the forty-five page product plan. Participants included the Vice Presidents of Development and Product Management, development team leads and systems engineers, selected programmers such as Ben, myself as usability lead, and two directors and two managers in product management. Discussions were stormy, fractious, and hostile. Strange alliances emerged from combined turf and ideological battles. Rick was livid at the plan's usurping of his control over features and priorities and rejected the plan despite its and his "program first" bias. Loyal to him, other "program first" developers joined the fray, charging cronies with a blatant attempt to take over. In response, cronies criticized "old Pyrrhtel" people for being too insular to realize that above all the venture needed to get a product to market and the plan would get us there. A series of major players unpredictably crossed factional lines, torn between their own inner interests and beliefs. Inwardly and outwardly, struggles were intense.

Eventually, the product plan was rejected, and the bitterness of the debates left badly battered egos and relationships. Product Management said that it would revise and resubmit the plan but it never did. This attempt to change organizational processes and officially get the group to buy into them was laid to rest. Neither workgroups nor authority was altered despite the recognized need to more deliberately plan, design, and develop market-driven products—in this case, an OEM product for the short term. Usefulness for end-users had little to gain from this particular plan but, in the long run, it may have benefited from instituting this new process of choosing features by sharing decisions and endorsements across the many roles present at these meetings.

The Vice President of Product Management's goal of taking control of both the product direction and the means for executing it in development did not disappear with the plan. Kevin continued but now more surreptitiously. He began working toward having Ben, the crony on the graphics team, supersede Rick as head developer venture-wide. Earlier, Ben had taken his frustrations in working with Rick and concerns about the direction of the graphics design to Stan, the VP of Development. Though sympathetic, Stan pronounced Rick "unmanageable" and made no changes in authority, status, or workgroup structure. Ben then took advantage of his close crony ties with the Kevin and Jack and went to them for support. Loathe to have this star crony disaffected, Kevin and Jack began orchestrating for Ben to head design and implementation and for Rick to take on more of a research role that would lack the power to thwart what the two executives jointly saw as essential "start up" talent.

From February through May, 1999, a sudden calm followed, fashioned more from people avoiding communication and repairing ruptures than from any sense of peace or resolution. Weary from the product plan confrontations, people went about their business and steered clear of conflict. The bulk of the developers worked on getting out the OEM graphics-and-platform toolkit by May. Three people were assigned to try yet another vertical product. But no one officially revisited whether it was time for the platform to move past the notion of overlapping needs between end-users and developer users.

During this period, six of us began work on a brainchild of Kevin's, the VP of Product Management—an attempt to woo investors by developing an add-in for Excel designed as a horizontal, generic visual problem-solving tool. Ben was named head of this project team, a position that began his rise in status and authority. He was given free rein to hand-pick five other user-centered advocates to join him. With him at the helm and calm giving us time to work, our horizontal product team moved freely in our own direction. We designed for usefulness, intent on circumventing established processes for feature requests that would thwart us from incorporating usability. If need be, we would build and modify the visualizations ourselves so that they supported and enhanced users' problem-solving processes. Revisiting the platform definition on our own, we decided to create our own platform features to serve the needs of end-users.

Breach in the Social Contract

Preparing and identifying user needs kept our horizontal product team busy through the middle of June, 1999. Around us, the venture went through its highest peak ever and, two weeks later, its lowest valley. The peak was the launch of the OEM product on schedule to venture-wide celebrations. The vale was the firing of fifteen people, a third of the venture. The reason for firing this time, Jack said, was the need to win investors. Venture capitalists were holding back because, they claimed, the venture was "too heavy to fly." The layoffs would make us sleek, fast, productive and saleable. Yet it was hard not to see that the CEO was also taking this opportunity to rid himself of detractors. A high-level director was fired who was one of Jack's and Kevin's most outspoken critics; most of the people let go were "old Pyrrhtel."

The venture reeled from this mass firing. With antagonisms still raw from the product plan and suspicions high about why we needed to find venture capitalists so soon, this crisis exacerbated mistrust, division, and fearfulness. Major players were lost from most teams; a sense of being adrift set in, and morale plummeted. In protest, one of the most highly regarded "old Pyrrhtel" developers resigned, a product manager equally respected by "start up" and "old Pyrrhtel" people alike. This time the "old Pyrrhtel" Vice President of Development did not try to buck up his troops. Genuinely pained at the costs of this firing, Stan became disenchanted with the growing backroom politics of the Vice President of Product Management and the CEO whose special agendas traced back to their tight relationship in their former company. This time, Stan wrote a two-page memo to the whole executive team warning that the layoff crisis and subsequent resignation revealed that the venture's unspoken social contract was breeched in ways that would be irreversible unless the executive team immediately renewed trust—renewed it by reorganizing workgroups, revitalizing them with a clear roadmap for a central product and its families, reinstating processes to abate the lack of direction, and opening decision making so that secrecy and privileged agendas no longer impeded communication and shared values.

Renegades

June, 1999, passed, however, and no moves came from the executive team to repair the damage. Teams struggled to regroup on their own but randomness prevailed. Our horizontal product team was the only team in the venture that didn't lose someone in the layoff. We coalesced strongly in the face of the surrounding disarray. Isolating ourselves from the turmoil fed into our already established tendencies to keep a low profile while intensely working to innovate the visualizations and their development processes. We created altogether new user-centered interface controls, graphics functionality, and infrastructure to suit users' demonstrated needs. We designed and built rapidly, and iteratively tested usability in the field. We sidestepped official processes to free ourselves from dependencies on "program first" teams. We evaded having to formally request and justify resources by informally securing them from Kevin, the VP of Product Management. In renegade style, we shared with the rest of the venture enough of what we were doing through our usability test reports and project updates to be good team players but our pace quickly outstripped our reported news.

We worked like this through July, 1999, but in November we hit a showstopper. In field-testing our product, users came to a halt in their analysis because one of the graphics invented by the Pyrrhtel researchers confused them. It was a particularly complex graphic, and no fix was possible in the short run. We had little time because we needed to coordinate with the marketing of Excel 2000 and get our product out in two months, no later than mid-October, 1999. At this point, discussions ensued with the full development and product management groups as well as with the executive team. Heated debates focused on whether to hold up production for three reasons: to improve the graphic, to assure that no other surprises lurked, and to take the time to follow all formal development processes so that we'd be selling quality software, not a prototype. "Old Pyrrhtel"—especially Stan, the Vice President of Development—insisted on following these formal processes and the wisdom of experience they represented. "Start up" and "user first" advocates wanted nothing to stand in the way of finally getting a product to end-users after a year and a half of trying. Members of these factions argued that continued improvements and quality could be added in a quick-turnaround Version 2 based on users' feedback from actual work contexts. "Program first" people were unnervingly quiet, seemingly hushed and alienated by how drastically this horizontal product deviated from their current work. To resolve these debates, the horizontal team proposed taking a month to invent and test an alternate approach to the same analysis with a different graphic. We would let users' demonstrated performance determine whether to go to market or not. Usability and usefulness would have an unprecedented final word.

Kevin and Jack—the VP of Product Management and the CEO—heartily concurred. Stan, however, demurred, sensing chaos. The rip in the social fabric that he had warned about had never been mended. This renegade approach seemed to threaten that contract even more. Though he endorsed the positive outcomes of the renegade team, Stan feared that products and processes were deviating in ways that threatened unity and coordination across products. For instance, the formal series of technical reviews by a larger team of developers and testers that had been standard Pyrrhtel practice had been shortcut. Teams were going their own ways and divisions were deepening. Stan was not satisfied with the revise-and-test solution because of the precedent it would set for quality assurance. He staunchly supported user-centeredness, more than the Kevin and Jack combined, but believed—as it turned out, accurately—that the commitment to usability in this case was being misused to tip the balance of power.

A month later, in September, 1999, the new design of the horizontal product was completed; field-testing showed that is was useful, usable, and valuable. It accomplished critical usefulness innovations that we had been trying to achieve for over a year. With this evidence, I backed going to market in October with our first end-user product. The launch date was set, and Kevin crossed Development boundaries to call for system testing. Despite the apparent success in revising the product, Stan balked. He was not against the product or its launch but against how it happened. The essence of this conflict captures a vexing struggle—the need to adequately answer the question "why back quality or usability if you can sell something without it?" Stan was backing the standard processes that ensured quality control even when, in this particular case, it was possible to get by without them.

He saw no recourse after this last incursion into his domain but to present Jack, the CEO, with an ultimatum. He demanded that Jack and

Kevin acknowledge his control of development and decisions about when future products were ready. Instead Jack fired him. In addition, he fired four other contrary "old Pyrrhtel" people, including one who, like Stan, was a founding father of the venture.

Aftermath

The innovative, user-centered product went to market two weeks later in October, 1999, on schedule. But the spirit and energy that gave rise to the possibility of creating this innovation were gone, gone with the disappearance—the silencing—of conflict. In November, 1999, in a desperate move, one of the executives, the last remaining founding father, went to the Board of Directors in an attempt to rein in the CEO and the Vice President of Product Management. This attempt failed. He and another executive subsequently resigned. Finding it no longer tenable to work in the current atmosphere, seven other people gave notice, including the last of the female programmers in the development group. Included in the group who quit were four out of the six people on the horizontal team who had brought the innovation to market, including me.

As far as I was concerned, within weeks after the firing, I saw that Stan's departure brought about a sea change in culture that was not going to sustain continued usability innovations. In the aftermath, upper management congratulated itself on its user-centered product but made no concrete plans or material commitments to foster more advancements in the next set of products. I was offered a promotion to Director and received assurances that the user experience would be important to future endeavors. But I saw too vast a discrepancy between these promises and the reality of how the executive team went about its business. Process had become a pejorative word, leaving little mechanism for continuity other than the whims of decision makers or the power brokering of interest groups. The resulting environment seemed to stifle the openness of debate that had stimulated creativity. The leaders preferred eliminating the sources of friction.

The firing brought into sharp relief a clear pattern—Jack who was hired almost two years earlier to do whatever it took to turn out a commercial product was intent on "cronifying" the venture and forcing out anyone who objected to his game plan. His game plan was not to turn out a revolutionary product but rather to rapidly turn out a "good enough" product that would interest venture capitalists and, ultimately, make the company an attractive purchase for a software giant like Microsoft. Usually we tend to think of a dualistic tension between process and product as, at first glance, may seem to be the case in Stan's conflict with Kevin and Jack. Stan had promoted process, and Kevin and Jack had made process out to be anachronistic and, ultimately, superfluous. But, in truth, Jack's game plan was based on a particular weighting in the triadic tension between process, product, and profit. Truly committed to neither product nor process he was ready to use and discard both in order to position himself for profit.

The effects of losing the Vice President of Development cannot be overemphasized. He was an ardent usability champion at the executive level. More importantly, in this multi-faction environment, he was one of the few people capable of taking on the crucial role of intermediary (Casson, 1997). He raised difficult questions, got disputing sides to enter into dialogue, and inspired them to take new perspectives for the good of a high quality, usable product. He knew how to engender and exploit cognitive dissonance for learning. Turbulent as many debates were, they often led to inventive designing and programming because, under the Vice President of Development's tutelage, people recognized the legitimacy of opposing positions and embraced the challenge of addressing them in the product. Stan's gift in facilitating this process went unnoticed—basically, taken for granted—until he was gone. After he was fired, the stark silences that occurred in meetings when difficult questions arose gave unsettling evidence of a significant loss of leadership.

The Politics of Innovation Revisited

On the face of it, this story looks like a struggle primarily between "user first" and "program first" with usefulness—"user first"—still being attained through renegade means before a series of "program first" definitions altogether closed out this possibility. But actually, in the long run, the most important factional dispute for usability was "start up" versus "old Pyrrhtel." The mythic quest of leaving behind stodgy R&D mentalities was enacted by the "start up" CEO and the Vice President of Product Management as they used usability as a weapon to oust the "old Pyrrhtel" Vice President of Development and almost all of the core Pyrrhtel people who had started with the venture. Unintentionally, "user first" principles figured into the firing of their best champion, which meant that repeated breakthroughs in usability became unlikely.

From a broad perspective, the usability experiences highlighted in this case history occurred in a larger framework that characterizes much of the software world today. Similar to this venture, many development contexts are in the process of moving from technology-driven to market- or user-driven products (Casson, 1997; Agre, 1997). The transition is difficult for the same kinds of technical and organizational reasons as we see in this case history. Inescapably, making the transition triggers conflicts between old and new ways of thinking about, designing and testing software. It brings together divergent cultures (Casson, 1997). One culture—an optimistic one—is ready to explore unknown markets, gather information about them and their users, stake out and design for these markets, and take risks. Another more skeptical culture prefers staying with familiar and known markets, gathering information only about new features that familiar and safe users may need, and averting risks. For this culture, the wisdom of experience and quality is embedded in current development processes, and neither the wisdom nor the processes should be abandoned. These cultures and their manifold variations all represent legitimate positions, which is what makes conflicts so rich, unity so elusive, and innovation so fragile.

Software firms and teams have many options for negotiating these conflicts and frictions, and the options that they pursue influence the content, scope, and structure of the programs they produce (Grudin, 1991). For most of my time in the venture, it seemed that the option was going to be to work with the tensions in constant dialogue. For a year and a half, the Vice President of Development was the much needed intermediary who integrated the "user first" and "start up" market-driven vision with "program first" and "old Pyrrhtel" concerns about familiar approaches and quality production. In addition, throughout the two years, Stan upheld the original vision of the visualizations transforming problem-solving. As time went on and upper management's decisions increasingly suggested only a raw profit motive, Stan served as a needed symbolic leader, keeping unity intact around a shared vision as well as he could (Casson, 1997). His leadership created the social atmosphere needed for cultivating mutual respect and negotiating deep-seated differences. But the product lagged in market-driven, end-user qualities.

As demand increased for the venture to be market-driven, the approach to dealing with tensions changed. At first tacitly and then overtly, the CEO and Vice President of Product Management "cleaned house." Negotiations gave way to formal power—most dramatically in the firing of people. Trust declined; and various groups struck out on their own to find some way to re-create trust and commitment. For some it was a heightened allegiance to rules and processes; for others it was the creation of a tightly knit renegade team (Casson, 1997; Thomas, 1994; Frost and Egri, 1995). The technology produced during this time had breakthrough usability innovations but the social atmosphere following the product release no longer tolerated dissension and a consultative mode of negotiating differences. Without this respect for differences, commitments to usability became readily expendable when deemed no longer useful.

A prime challenge for any development context that is undergoing a transition from system- to user-centered products is to figure out how a common culture and conflict can coexist. A common culture is necessary for sharing a motivating vision, coordinating a wide range of talents around innovation, and gathering and exchanging information across a large enough network to inspire repeated innovation achievements (Casson, 1997; Schein, 1996). Building a common culture to support innovation is a long-term endeavor. Renegade strategies, by contrast, provide only a stop gap measure. They are often the main strategy found in the literature about success in technological innovation but these renditions are usually project-based, ending them with innovative outcomes not aftermaths (Thomas, 1997, Frost and Egri, 1995; Sharrock and Button, 1997; Law and Callon, 1995). Renegade efforts sprint to success but sprinting does not work for a marathon.

In any environment that is making the transition from technology- to market-driven products, innovations incite competing agendas and interests expressed in highly charged politics. Usability can easily become a pawn in political games, dispensed with as readily as it is embraced. It may seem to be a feat to achieve usability innovations in a project but that single success does not guard against usability becoming dispensable. The technical accomplishments may be fleeting if they have not been grounded in organizational processes that engender trust and constructive debate and that strive for an equitable balance among profit, product, and process. Usability specialists have a crucial organizational and political role to play in bringing out these processes. We cannot assume that if we simply educate development and product management teams about usability the culture and its processes will change accordingly. Political stakes in transitional environments are too high for such idealistic hopes. Rather we need to actively promote organizational as well as technological innovation. Promoting organizational change, in part, involves knowing enough about usability in the earliest stages of a product's life cycle to dissuade developers from making choices about architecture and scope that will foreclose opportunities later to design for usefulness-in-context. Equally important is gaining a seat at the decision-making table and informing the choices that we make with an awareness of long- and short-term goals, of intended and possible unintended consequences. From positions of influence, we can work toward achieving structures in workgroups, arrangements of power, and modes of decision-making that produce consultative, flexible, and trusting environments. These environments provide the deep structures necessary for usability to take hold and flourish.

This case history underscores that overt resistance is not the only threat to usability. In subtle but intricate ways, overt support also may set back the cause of usability considerably if that support is strictly expedient rather than authentic. With this insight in mind, it becomes vital for us to maneuver politically for genuinely supportive environments and to embed usability deeply into them. For usability to endure, it takes political as well as technical skills and knowledge.

Acknowledgements

I would like to thank Russell Borland and Roger Theodos for their insightful comments and extremely helpful advice on drafts of this article.

Appendix A: Timeline of Events

December 1997	Visible Solutions is formed as a Pyrrhtel venture.
January-April 1998	Eight new cronies are hired.
January 1998	Ben and Rick disagree about Active-X design and library.
	Factions arise between "start-up" and "old Pyrrhtel."
March 1998	Pat becomes the lead of the Platform team.
April 1998	Six Pyrrhtel people are fired.
May 1998	The Platform architecture is defined.
	Debates give rise to "program-first" and "user-first" factions.
June 1998	Four development teams are underway.
	Rick and Pat are in charge respectively of graphics and platform priorities for features.
July 1998	Vertical teams submit user requirements and user-centered requests for feature fixes and enhancements.
	Disputes arise over how to decide on features and priorities.
	Rick introduces and implements the chit system.
December 1998	The first vertical team folds.
	Users for the second vertical team's product give feedback on usability problems.
December 1998—January 1999	
	The second vertical team redesigns its product.
	The second vertical team's partner company becomes troublesome.
End of January 1999	The second vertical team folds.
	Product Management issues a 45-page product plan and feature list.
	Three days of meetings occur to discuss the product plan and feature list.
	The product plan and feature list are abandoned.
	Jack makes his first foray into getting venture capital in order to spin off from Pyrrhtel.
February—May 1999	Calm prevails. Platform and graphics teams continue.
	A new vertical team is formed.
	A new horizontal/generic product team is formed; it will become the renegade team that implements innovations for usefulness. Ben becomes team lead.
May 1999	The OEM product is launched.
June 1999	Fifteen people are fired. Another quits.
	Stan writes a memo to the executive team warning them of a breech in the social contract and a venture-wide crisis. No action is taken.
July 1999	The horizontal product team pushes ahead with its usability innovations.
November 1999	Users hit an obstacle with the horizontal problem-solving product.
	Re-design and testing occurs.
	Usability is given the final word on whether to ship or not after testing.
	Stan objects; Jack and Kevin concur.
September 1999	The new design passes usability tests without problems.
	The product is prepared for shipping in October.
	Stan demands greater control over development decisions. He is fired.
	Four other "old Pyrrhtel" people are fired.
October 1999	The horizontal product ships on schedule.
November 1999	One of the remaining Vice Presidents appeals to the Board to rein in Jack and Kevin; the appeal fails.
	That VP and another resign.
	Seven other people in the venture resign.
December 1999	Visible Solutions gets investor backing and spins off from Pyrrhtel.

References

Agre, Philip (1997). *Computation and Human Experience*. Cambridge: Cambridge University Press.

Albers, Michael (1999). Information design considerations for improving situation awareness in complex problem-solving. In *Proceedings of the Seventeenth Annual Conference of Computer Documentation* (pp. 154-158). New Orleans: Association for Computing Machinery.

Anderson, R. J. (1994). Representations and requirements: The value of ethnography in system design. *Human-Computer Interaction,* 9, 151-182.

Beyer, Hugh and Karen Holtzblatt (1998). *Contextual Design: Defining Customer-Centered Systems*. San Francisco: Morgan Kaufman.

Brown, John Seely and Paul Duguid (1991). Enacting design for the workplace. In Paul Adler and Terry Winograd (Eds.), *Usability: Turning Technologies into Tools* (pp. 164-197). New York: Oxford University Press.

Casson, Mark (1997). *Information and Organization: A New Perspective in the Theory of the Firm*. Oxford: Oxford University Press.

Cooper, Alan (1999). *The Inmates are Running the Asylum*. Indianapolis: SAMS.

Coyne, Richard (1995). *Designing Information Technology in the Postmodern Age: From Method to Metaphor*. Cambridge, MA: MIT Press.

Czerwinski, Mary, Maarten van Dantzich, George Robertson, and Hunter Hoffman (1999). The contribution of thumbnail image, mouse-over text, and spatial location memory to web page retrieval for 3D. In M. Angela Sasse and Chris Johnson (Eds.) *Human-Computer Interaction INTERACT '99* (pp. 171-178). Amsterdam: IOS Press.

Denning, Peter and Pamela Dargan (1996). Action centered design. In Terry Winograd (Ed.), *Bringing Design to Software* (pp. 105-119). Reading, MA: Addison-Wesley.

Feenberg (1995). Subversive rationalization: Technology, power, and democracy. In A. Feenberg and A. Hannay (Eds.), *Technology and the Politics of Knowledge* (pp. 3-22). Bloomington, IN: Indiana University Press.

Forrester Research (1998). *Why Most Web Sites Fail*. Forrester Research, Inc: September.

Frost, Peter and Carolyn Egri (1995). The political process of innovation. In Cynthia Hardy (Ed.), *Power and Politics in Organizations*. Brookfield: Darmouth Press.

Giddens, Anthony (1984). *The Constitution of Society*. Berkeley: University of California Press.

Grudin, Jonathan (1991). Interactive systems: Bridging the gaps between developers and users. *Computer,* 24 (April), 59-69.

Holland, John (1998). *Emergence From Chaos to Order*. Reading, MA: Perseus Books.

Jackson, Michele (1996). The meaning of "communication technology." In Brant Burleson (Ed.), *Communication Yearbook 19* (pp. 229-268).Thousand Oaks, CA: Sage.

James, Janice (1999). Frequently tested issues in usability studies in websites. In *Proceedings of Usability Professionals Association* (pp. 451-462). Scottsdale, AZ: UPA.

Johnson, Robert (1998). *User-Centered Technology*. Albany: SUNY Press.

Johnson, Peter, Hilary Johnson, and Stephanie Wilson (1995). Rapid prototyping of user interfaces driven by task models. In John M. Carroll (Ed.). *Scenario-Based Design* (pp. 209-216). New York: John Wiley and Sons.

Johnson-Eilola, Johndan (1997). *Nostalgic Angels: Rearticulting Hypertext Writing*. Stamford, CT: Ablex.

Knights, David and Fergus Murray (1994). *Managers Divided: Organizational Politics and Information Technology Management*. Chichester: John Wiley and Sons.

Latour, Bruno (1988). The *prince* for machines as well as for machinations. In Brian Elliot (Ed.). *Technology and Social Process* (pp. 20-43). Edinburgh: Edinburgh University Press.

Law, John and Michel Callon (1995). Engineering and sociology in a military aircraft project: A network analysis of technological change. In Susan Leigh Star (Ed.), *Ecologies of Knowledge* (Ch. 8). Albany: SUNY Press.

Liddle, David (1996). Design of the conceptual model. In Terry Winograd (Ed.), *Bringing Design to Software* (pp. 17-31). Reading, MA: Addison-Wesley.

Nardi, Bonnie (1996). *Context and Consciousness*. Cambridge, MA: MIT Press.

Nielsen, Jakob (2000). *Designing Web Usability*. Indianapolis: New Riders.

Norman, Donald (1999). *Invisible Computer: Why Good Products Can Fail, the Personal Computer is so Complex and Information Appliances are the Solution*. Cambridge, MA: MIT Press.

Nyce, James and Jonas Lowgren (1995). Toward foundational analyses in human-computer interaction. In Peter Thomas (Ed.). *The Social and Interactional Dimensions of Human-Computer Interfaces*. Cambridge: Cambridge University Press.

Ramey, Judith, Alan Rowberg, and Carol Robinson (1996). Adaption of an ethnographic method for investigation of the task domain in diagnostic radiology. In Dennis Wixon and Judith Ramey (Eds.). *Field Methods Casebook for Software Design* (pp. 1-16). New York: John Wiley and Sons.

Rheinfrank, John, William Hartman, and Arnold Wasserman (1991). Design for usability. In Paul Adler and Terry Winograd (Eds.), *Usability: Turning Technologies into Tools* (pp. 15-40). New York: Oxford University Press.

Rubin, Jeffrey (1994). *Handbook of Usability Testing*. New York: John Wiley and Sons.

Schein, Edgar (1996). Culture: The missing concept in organization studies. *Administrative Science Quarterly,* 41, 229-240.

Sharrock, Wes and Graham Button (1997). Engineering investigations. In G. Bowker, S. L Star, W. Turner and L. Gasser (Eds.), *Social Science, Technical Systems, and Cooperative Work* (pp. 79-104). Mahwah, NJ: Lawrence Erlbaum Associates.

Spool, Jared, Tara Scanlon, Will Schroeder, Carolyn Snyder, and Terri DeAngelo (1999). *Web Site Usability: A Designer's Guide*. San Francisco: Morgan Kaufman.

Star, Susan Leigh (1995). The politics of formal representations: wizards, gurus, and organizational complexity. In Susan Leigh Star (Ed.), *Ecologies of Knowledge* (pp. 88-118). Albany: SUNY Press.

Thomas, Robert (1994). *What Machines Can't Do: Politics and Technology in the Industrial Enterprise*. Berkely: University of California Press.

Wiklund, Michael (Ed.) (1994). *Usability in Practice*. Boston: AP Professional.