CONTEXTUAL DESIGN

Defining Customer-Centered Systems

HUGH BEYER

KAREN HOLTZBLATT

INCONTEXT ENTERPRISES

Morgan Kaufmann Publishers, Inc.
San Francisco, California

Introduction 1

Developing software has never been easy. But over the last 20 years the requirements for software development have gotten far more stringent. Once computers were used by experts in glass rooms; now everyone on the street expects to use a computer to get their jobs done. Once computer users knew and liked technology; now users want their computers to be as invisible as a ballpoint pen so they can focus on their jobs. Once applications supported a single, bounded task—compute compound interest for a bank's loans, perhaps; now they are expected to support the whole work of the business, from electronic funds transfers with the Federal Reserve to the company's email system. It's no longer enough to be a good software engineer. To be successful in today's world, those who define and build hardware and software systems[1] must know how to fit them into the fabric of everyday life.

Commercial software vendors recognize the reality of the new situation when they emphasize "solutions" over products. Traditionally, new products were most often defined by an engineer getting a bright idea, building it, then looking for a market for it. But the new demands of the market suggest that the new product won't be accepted if it doesn't fit with customers' other systems and existing ways of working. Customers are looking for an integrated set of products that solve whole work problems, not point products that don't work with anything else, that don't seem to solve the problems they have, and that are too hard to use.

The challenge of system design is to fit into the fabric of everyday life

Commercial product vendors are one major segment of the software development industry; the other is the Information Technology

[1] For simplicity's sake, we use "system" to refer to any combination of hardware and software used to deliver a product, application, or computer platform.

(IT) departments, building the systems that run a company. Their customers[2] are the people actually doing the work of the business. The new user expectations have hit IT departments just as hard as commercial vendors. Taught by the ubiquitous desktop systems, their customers expect that all systems will be as easy to use. They expect to be able to access and manipulate corporate data from the PC on their desk as easily as they access their own desktop files. When they can't get the systems they want, these customers decide that the IT department is out of touch, has far too long a response time, and too often delivers systems that can't be used. Then the customers change direction, cancel the IT project, and buy some desktop solution off the shelf—which they expect the IT department to maintain. The IT department responds with processes designed to help them manage the demands on them: Joint Application Development (JAD) sessions to clarify requirements, formal sign-off to control changes, and enterprise modeling to recapture some of the initiative. But in the end it's the customers' system, and there's a limit to how much these processes can keep them from changing their minds.

The problems both kinds of organization are struggling with have the same root. Requirements engineering—front-end design—systems analysis—whatever the term used to describe the activity, the hard underlying problem is determining what to build to help people do their work better and specifying it at a level of detail that developers can code to. Customer-centered design promises a solution, but taking advantage of it leads quickly to questions about the nature of systems development and the organizations that practice it. What is the right way to define new systems? What's the relationship between those who say what to build and those who build it? How do we make sure the system specification defines something the customers really want? And how do the different parts of an organization work together to invent and deliver

Contextual Design is a backbone for organizing a customer-centered design process

[2] We'll use "customer" to refer to anyone who uses or depends on a system—it's a more inclusive term than "user," which we'll use only for those who interact with the system directly. There's some dispute as to whether the "customer" or the "user" should be primary in the design process. Some worry that the term "customer" leads the design team to focus too much on those who pay for a system, rather than those who use it. We recognize that danger, but also recognize that a system must meet the needs of *all* those who depend on it, and so prefer the more inclusive term.

a coherent system? An approach to system design that hopes to have an impact on real organizations must be able to answer these questions. (See "Readings and Resources" for additional perspectives on the problem.)

Contextual Design (CD) is an approach to defining software and hardware systems that collects multiple customer-centered techniques into an integrated design[3] process. Contextual Design makes data gathered from customers the base criteria for deciding what the system should do and how it should be structured. It makes deciding how customers will work in the future the core design problem and uses those decisions to drive the use of technology. It unifies all an organization's actions into a coherent response to the customer. And it defines activities focused on the customers and their work, rather than leaving team members to argue with each other based on personal opinion, anecdotes, or unverifiable claims about "what customers would like."

THE CHALLENGES FOR DESIGN

Making customer-centered design practical for real engineering organizations depends on striking a balance among multiple considerations. For customer-centered design to be possible at all, the process needs to include techniques for learning about customers and how they work. This means that we must discover the everyday work practice of people. But anyone's real work practice is intricate and complex; understanding it in depth leads to an overwhelming amount of immensely detailed information. One typical response to such large quantities of data is to "reduce" it—perhaps by summarizing the top five issues in all the data and just responding to those. Another typical response is to decide that the problem is too big to address—and instead to deal with one customer

Collect and manage complex customer data without losing detail

[3] Throughout this book, we use "design" in the ordinary English sense of conceiving and planning a system. This is how Mitch Kapor uses the word (Kapor 1991). The technical software engineering usage of "design" is different; it applies to the design of the implementation only. Since our topic is customer-centered *design*, we have reappropriated the term; Chapter 11 discusses how this activity fits into the development life cycle.

problem or issue at a time, respond to that one issue, and ignore the rest. One team found 100 different user needs, grouped them into 20 application areas, and assigned each to a different team—resulting in 20 unintegrated point solutions. None of these approaches give the design team the ability to respond to the customers' whole work practice with a coherent set of systems. The trick is to give the team tools that let them see the breadth of data without being overwhelmed, to see the common structure and pattern without losing the variation, and to understand the wealth of detail without losing track of its meaning.

Seeing customer data is critical, but so is understanding how to design a response. In customer-centered design there are three levels of design response that matter. First, and most important, is the design of work practice. If the team is to define a new system that fits into the fabric of its customers' lives, then the team—in partnership with the users themselves—needs to see and redefine that fabric. This allows them to define a new work life that hangs together for the user. Second comes the design of the corporate response that delivers the new work practice. It is not enough to design the system alone. A custom software system's internal users need to integrate organizational roles, business procedures, and the system (perhaps including software, hardware, and communications connections) that supports them. A commercial product depends on the definition of the market message, associated services, delivery mechanisms, and the product itself. All these different aspects must be planned and delivered together to create a viable business. Third, when the nature of the corporate response has been defined at a high level, the team can design the structure of the system itself. Whether software only or software and hardware combined, the system creates an environment for its users to work in; it's up to the team to ensure that that environment fits the flow of their work. The challenge for customer-centered design is to provide for all three levels of design in a process that guides the design team's daily actions and fits within the constraints of the organization.

Design a response that is good for the business and the customers

It's people who create a design and people who have to work together to make it happen. Putting a team to work on a problem, rather than one person, means an organization can handle larger and more complex problems. A team has multiple skills and points of view to bring to bear on a problem. A cross-functional team drawn from the departments that have to cooperate to produce a system can account for the issues and needs of each department. Designing effectively together depends on techniques that manage the interaction between people in a room so that they can create a unified corporate response. But pulling a cross-functional team together means breaking down some of the walls between parts of the organization. Given the extreme pressure all organizations are under to produce results quickly, the different groups must be able to work in parallel once the corporate response is defined, while still maintaining the coherence of the total effort.

Foster agreement and cooperation between stakeholders

Contextual Design deals with the issues of gathering data, driving design, and managing the team and organizational context. It has evolved over the last 10 years through intensive work with teams producing products and internal systems, and designing organizational processes. This grounding in real experience has ensured that Contextual Design takes the needs of working design teams into account, providing methods to develop insights and shared direction among team members at each point. Contextual Design provides complete support for the design process, from the initial customer data gathering through the transition to object-oriented design (or whatever other implementation model you favor). The process brings together the techniques needed to design a system that meets its customers' needs, while addressing the challenges of making a design process work in real-world situations.

Make the process practical given real time-driven organizations

THE CHALLENGE OF FITTING INTO EVERYDAY LIFE

Federal Express has changed how businesses work on a daily basis by providing an affordable, reliable way of getting packages to another location overnight. Spreadsheets have made elaborate numerical models commonplace, where once they were the domain of a specialist. And even such a simple thing as the mouse and windowing user interface (UI) has helped move computers from specialized tools to an integral part of everyday work. These products and services are important because they make new ways of working possible.

These examples suggest that the critical aspect of a new product or service is the new way of working that it enables. But what does it mean to enable a new way of working? How is it that a system might support or disrupt work practice?

Consider the true story of one user trying to do a simple task: A user of a standard office system needs to print a label. From her point of view, this should be simple. She should write her letter, putting the address at the top. She should tell the computer to print an address label, and put a label in the printer. The computer should get the address from the letter and print it on the label. This is her *user work model* for the task: these are the concepts she uses and the strategy she takes for granted to get her label printed. If they were built into her systems, her work would be straightforward.

Instead, the system offers its own *system work model,* which imposes new distinctions and a style of work foreign to her. She writes the letter, with the address at the top, but she can't just print the label from that address. First, she must create a separate document containing only the address so it will print properly on her labels. She copies the address from the letter to this separate document. She tells the system to print the document, remembering to say that it should use the manual feeder. It will use the sheet feeder if she forgets, even if she's put a label in the manual feeder (it's easy to forget). She waits for the system to realize that it has to ask for the next sheet. (She must not insert the label before the system asks, or the printer will spit it out without printing on it.) When the dialog box comes up saying the "print manager" (the what?) has a problem, she dismisses the dialog box and switches to the print manager application. She cannot tell the print manager to continue without switching to it. She goes to the printer (which is across the room), takes it offline, inserts the label, and puts the printer back online. (If she inserts the label without taking the printer offline, it spits it out without printing on it.) She goes back to the computer and dismisses the dialog box requesting a new sheet of paper to print on. (It will not sense that she put a label in.) Finally, she switches back to her application.

Support the way users want to work

Printing a label is a conceptually simple task. But this system presents an enormously complex model for it. It introduced many new steps, driven by the concepts of technology, not by the needs of the work. It introduced new concepts that the user must understand to get her job done: What's the print manager, and how is it different from an application? How do you switch applications? If the print manager can put up a dialog box asking the user to switch to it, why can't it also say what's wrong? If the system can tell there's no label in the printer, why can't it tell when there *is* a label in the printer? What's the distinction between on- and offline? How is that different from on and off? The system's model is hard to understand because it makes no sense in terms of the work people are trying to do. And the net result, of course, was that this user never did use the computer to print labels. The new work practice was so foreign and cumbersome she preferred to continue writing labels by hand.

This system supports work poorly. It is poor not because functions are missing but because the system imposes a work model that does not make the job more efficient and does not match the user's expectations. The designers of this system had no choice about imposing a way of working. *Any* system imposes a model of work. The only choice designers have is whether they will design that work model explicitly to support the user or whether they will allow it to be the accidental result of the technical decisions they make. This model of work was created by the interaction of multiple tools designed by multiple groups in several organizations. No job, even one as simple as printing a label, is accomplished with a single tool. Design has become difficult because systems now support almost every aspect of work life. It's up to the design team to understand the environment their tool will be used in because it is the combination of tools that controls their customers' work practice. (See Terwilliger and Polson [1997] for related research.)

Don't increase work and frustration with automation

CREATING AN OPTIMAL MATCH TO THE WORK

Does this mean that a new system must match its customers' existing work practice exactly? Certainly not—that would be a sure path to failure. But systems must match the user's model closely enough that the user can make the transition. History is littered with excellent innovations solving real problems that have never been adopted because it is simply too hard to switch models. The Dvorak key layout *will* increase your typing speed—if you are willing to retrain your fingers and accept incompatibility with virtually every keyboard in existence. Switching to DC current in your home *will* eliminate the risk of electrocution to you and your family—if you are willing to install a converter and replace every appliance you own. Neither idea has gone anywhere because the cost of change is too high.

Then how can a design innovate successfully? By taking one step at a time, always considering the interaction between the new ideas and the current work practice. Consider the history of the word processor. Originally, everyone used typewriters, and typing became the work model users understood. Early word processors stayed close to the typewriter model. They just provided better typing and better correction. Then word processors introduced cut and paste—metaphors taken from the physical operations of cutting with scissors and pasting with glue, something everyone had to do already. These features were an easy extension of the model. Then word processors introduced multiple buffers and multiple documents open at a time, making it easy to share and transfer text across documents. Then they introduced automatic word-wrapping and multiple fonts, and desktop publishing was born. Each step was an easy increment over the previous, and each step walked the user community a little further away from the typewriter model. Now word processing has little but the physical act of typing in common with using a typewriter.

Innovate through step-by-step introduction of new work practice

The history of word processing illustrates how work can be revolutionized over time. A good design provides an *optimal* match between the users' current way of working and the work practice introduced by the new system; it changes the work enough to make it more efficient but not so much that people cannot make the transition. Innovative designs that succeed are those that offer new ways of working and new advantages while maintaining enough continuity with people's existing work that they can make the transition.

Determining what makes an optimal fit is a decision for the design team; there's no absolute right answer. It's part of the design process to decide how to integrate an innovation into the customers' work practice so smoothly that they can successfully adopt it. In customer-centered design, we seek a framework for the discussion so that the decision is based on customer data, and a way to check the decision with the customer.

Keeping in touch with the customer

If designing from an intimate understanding of the customer is so basic, why is it so hard to achieve? As product development companies grow, they create organizations that have the effect of keeping designers away from their customers. A start-up expects their developers to help make the sale by talking to potential clients. But, as it grows, it develops a sales organization to handle the customer interface; it puts account representatives in place to control the sales organization; and it puts marketing and product management organizations in place. All this tends to keep developers away from even the salespeople. A start-up puts developers on the customer support line. But, as it grows, a whole organization takes over the customer support function, with a formal interface for providing feedback to development. Developers are isolated from immediate customer feedback about how they are doing. We've talked to developers so isolated from their customers and so powerless to make changes that they didn't even want to talk to customers because they didn't think they could fix any problems they found.

Organizational growth isolates developers from customers

IT departments have difficulty staying close to their customers for a different reason. They, too, tend to become isolated from their customers as the company grows, but the different solutions available to them each come with attendant problems. They create new roles—business, systems, or requirements analysts—to translate between customers and developers, but find that customers still believe the IT department doesn't understand their business. The additional layer doesn't create a close working relationship with the customers, and it doesn't create a clean handoff to development.

Sitting with the users makes cross-departmental projects hard

To control shifting requirements, IT departments put sign-off processes in place, but customer priorities and requirements still change. Then they situate developers with the people in the businesses they support, so they are closely involved with the work of the business. This improves the client relationship, but it means the IT department can't share resources or expertise and can't take a strategic role looking across the whole company's information systems. So they decide they are too fragmented and pull everyone back together again, reintroducing the problems of isolation.

This kind of oscillation is typical in IT departments, but in the end it misses the point. Any arrangement of people comes with its attendant problems—the only solution is to recognize the problems and address them. The IT department needs some distance from the customers to see across the stovepipes created by the different departments and plan systems that address the business as a whole (along with a way to fund such systems). At the same time, they need mechanisms that keep them in close partnership with their customers.

This is the challenge for Contextual Design: to make a design team's understanding of their customer explicit and give them enough distance to see the work practice as a whole, across the business or across a market. Yet at the same time, the process must keep the design team thoroughly grounded in the knowledge of what's real for their customers.

THE CHALLENGE OF DESIGN IN ORGANIZATIONS

Who gets to say what a system will do? Is it really the marketing department or systems analysts saying, "Build this," with the engineering team just following their specification? Or do marketing or the analysts really only say, "Make this *kind* of a thing," with the engineers actually deciding what they will build and how the system will work? In fact, both sides have a role to play in saying what a new system will look like—the creation of a system in real organizations is the outcome of their cooperation.

The underlying problem is inescapable. Today's systems are too large to be built by a single person. So organizations divide the process of defining and building a system into parts and assign each part to a group of people. The people in each group specialize in their own part and lose contact with what all the rest are doing. There are four questions to answer in the course of developing a system, and these questions tend to define natural breaks in the development process, so it's easy to assign one group to answer each question. The questions are, *What matters in the work*—what aspects of work should be addressed? *How should we respond*—

Breaking up work across groups creates communication problems what kind of a system should a team respond with? *How should the system be structured*—what exact function, arrangement of function, and system structure best meets the needs of the work? And *how are we doing*—does the system as designed actually work for the customers?

The first question (what matters?) asks what aspects of the customers' work practice a new system should address, what issues or problems should be overcome, what roles and tasks are important to support. The group tasked with answering this question is typically marketing or business analysts. When management changes an organization, they often define what matters for its information systems in directives. "Too much overhead goes into approving purchases," they say. "Give every group a credit card and authority to use them." Or, "Our chemical databases are our lifeblood—tie together all the databases across the company." Or marketing might tell their engineering group, "Design a product to support business planning," or "Put this product on the Web." These directives say at a very high level what work issue the system should address, but don't really define the system. What aspects of a product are affected by putting it on the Web? Does tying databases together mean one database, replicated databases, or a way to search across multiple databases? From the point of view of those setting direction, these questions are details; it's not their job to answer them.

Different organizational functions focus on different parts of a coherent process

Answering these questions means saying what the response will be: how the corporation will coordinate to respond to the issues with system designs, processes, services, and delivery strategies. Marketing may be part of defining the response unless the company is very engineering-driven. Requirements analysts may do this unless they are very nontechnical, in which case developers will drive it. The customers themselves should be involved with an internal system, since it defines how they work. A research or architecture group may drive defining a response. Marketing and analysts may have the formal charter to "develop the specification for the system," but in our experience they don't really define system behavior to the level of detail needed to write code. "They give us a specification," the developers tell us. "But there's always lots that we have to decide, and they usually ask us for things that are really impractical. So it's give-and-take."

Deciding how to structure the system means deciding exactly what function to include, exactly how the system will behave, and how it will appear on windows, menus, or screens. This is nearly always done by developers, which means they need to understand the whole work context in which the system will be used. Otherwise they cannot make decisions that are appropriate for the customer. Developers don't get customer data at this level of detail from their marketers or systems analysts. Working out the detailed system structure depends on an additional level of customer data that developers have to get themselves—or design the system based on what feels right to them. Some companies have gone so far as to put the development group in a different state from their users and analysts, to create a group focused on its development work. But that just creates a greater need for communication and causes more serious isolation when travel budgets are cut.

Every function needs customer data, but it has to be the right kind of data

The final question for design asks, How are we doing? This question checks the progress of the system with the customer to ensure that it's still the right system, and low-level changes haven't made it unusable. This question is often separated out and given to a usability or Quality Assurance group to test. Answering the question looks at the system itself (for bugs and fit to the specification) and tests the system with users. But in either case, dealing with the results is the job of the developers. So they have to receive, understand, and believe in the feedback—which means they have to buy into the process of getting it and trust the group that collects it. And often, when there's been no real design from data so far, the flaws discovered at this point are so fundamental they cannot really be addressed at this stage of development.

Data showing what is wrong is frustrating if builders can't fix it

All these different parts of defining what a system will be have to come together if the system and design process are to work. The people defining the response have to respond to real work problems; the people building the system have to build the response they agreed to. But keeping each part isolated to its own group creates communication problems across the organization. The formal documents in engineering processes capture the evolving design, but also are intended to manage the communication and disagreement between groups. "You signed off on this specification. That means you're committed to it." "Yes, but we've reorganized and don't need that anymore." Or, "Yes, but we talked to this important customer and there's just no point in shipping if we can't meet this need." Or, "Yes, but it's not possible to implement that given the technology we have."

Design in organizations is about developing a coherent direction across all the groups: agreement on the corporate response they intend to deliver. It's not that changes will never happen—it's that when changes happen, the whole organization can respond appropriately across all functions, rather than turning the changes into an argument between two groups. In turn, a coherent view depends on taking account of all the different perspectives during the development of the corporate response. Marketing and analysts need the technical perspective to see opportunities for new kinds of systems. Engineering needs the marketing perspective to see why some directions make a good product and others don't. They need the analyst's perspective to see the work issues they might address. An IT team needs the customer's perspective to ensure their proposed changes to working procedures are reasonable and will be accepted. After they've developed a corporate response, they can work in parallel for efficiency without losing the single direction. But in up-front design, a cross-functional team works best.

Cross-functional design teams create a shared perspective

TEAMWORK IN THE PHYSICAL ENVIRONMENT

Creating a cross-functional team that does design work together runs into some surprising problems in real organizations. Consider the most basic question: where is such a team to meet? A quick look at the physical structure of most organizations would make you think they are designed to keep people apart. The most common work environment for a developer is the cubicle—an area big enough for one person to work in comfortably, containing a terminal and a desk. But it is not big enough for several people to work together comfortably, and it does not have the wall space to support group work. Meeting rooms do exist, but a meeting room's key characteristic is that it is shared and booked by the hour. Because it is booked by the hour, the only work that it supports is that which can be completed in a short time—a half day at most. Start booking rooms for longer than that on a regular basis, and you get dirty looks

Organizations have no real spaces for continuing team work from your coworkers. After all, you are hogging a shared resource. Not only that, but because the room is shared, you can't leave much stuff in it. Every conversation has to restart from scratch, and every meeting has to start with spreading out all the design diagrams again.

So the only work a meeting room supports is work that can be completed in a few hours and that does not require much physical support—no hardware, no charts, no diagrams, nothing you cannot roll up in a few minutes and take with you. Maybe there's a network hookup—but to which LAN? And is it still good? And how do you hook this laptop PC to it anyway? Given these constraints, is it any wonder that designers and engineers consider meetings a waste of time? The very physical structure of a typical large corporation announces plainly that real engineering happens alone in cubicles and that when people gather in a meeting room, they are not doing real work. There's nothing in the usual structure of organizations to support the face-to-face needs of initial system design.

MANAGING FACE-TO-FACE DESIGN

Working together effectively means having workplaces where real work, done by multiple people working face-to-face, can happen. It also means giving these people the interpersonal skills and process to make their sessions effective. One division manager addressed a particular thorny problem by booking rooms at a local hotel, sending his five senior architects there, and telling them they were fired if they didn't come up with a solution in a week. He got a result, but we find people are usually happier, more creative, and produce results more quickly if they have a reasonable process to work in. The typical response to having a process—even from people who "hate" process—is, "Thank you. Now I finally know what to do."

Face-to-face work depends on managing the interpersonal

Working together is a new skill; it is not something taught in schools and is rarely taught on the job. Working together effectively means understanding how to keep a design conversation on track, how to focus on the work issue and not each other, how to manage everyone's personal idiosyncrasies, and how to uncover and address the root causes of disagreements. Unless teams learn to do this, their designs suffer because the models people have for handling disagreements trade off coherence of the design for keeping people happy.

One primary model people have for handling disagreement is horse trading: "I think you're wrong on that. But I'll let you have it if you'll give me this other thing that is really important to me." Horse trading leads to a system that is a patchwork of features, with no coherent theme. And horse trading causes everyone on the team to disinvest from the design because everyone has had to agree with at least one decision they thought was fundamentally wrong.

Other models for handling disagreement exist, but most don't work any better. There's the compromise model, which says, "You say we should design everything as dialog boxes. I think everything should be buttons. So we'll implement both and make everyone happy." Everyone is happy except the user, who has a dozen ways of doing each function and no clear reason to choose one way rather than another. Or there is the guru model, which says, "The guru is smart and knows everything. We'll all do what the guru says." That is fine, except the population of gurus who are infallible on technical architecture, GUI design, user work practice, marketing, project planning, and the host of other skills necessary to get a product out is vanishingly small.

Disagreements can lead to an incoherent design

Contextual Design defines a process for developing systems that takes the interpersonal issues into account. It defines procedures for deciding among design alternatives based on data, not arguments or horse trading. It defines roles for people to take on during design sessions that keep the discussion on track. It does this not only to make the design process more efficient, but because when people argue and have no process for making decisions, it pulls the system apart.

Customer-centered design keeps user work coherent by creating a well-working team

Front-end system design forces us to address interpersonal issues because it's in this part of the design process that bringing different functions face-to-face matters so much. Traditional design draws less heavily on the skills of working together. Committing to keeping the customer work coherent despite the different perspectives and skills on the team makes knowing how to work together critical. We find that when people have a clear process and clear roles to play, when they become sensitive to the individual styles that cause them to clash in the room, and when they have concrete data to base decisions on, they can overcome the barriers to working together effectively.

This is what it means to be customer-centered: not only does customer data drive design, but the design process leads to a system that keeps the users' work coherent in the system, from invention through implementation. The challenge for Contextual Design is to build in techniques that recognize the issues around working together and provide ways to do so effectively.

THE CHALLENGE OF DESIGN FROM DATA

Learn how to see the implications of customer data

Design is a cognitive activity. It is thought work. It begins with a creative leap from customer data to the implications for design and from implications to ideas for specific features. A clear understanding of the customer doesn't *guarantee* any kind of useful system gets designed and delivered. Design depends on being able to see the implications of data. In many of the classic stories about the development of new systems in the computer industry, inventors were responding to their unarticulated sense of what was important based on their own experience. Ken Olsen was an engineer building digital circuitry. He knew other engineers would buy smaller computers if they were available and invented the first minicomputers. Dan Bricklin learned accounting while getting his MBA. He knew accountants would use automated spreadsheets and invented VisiCalc. These entrepreneurs responded to their experience with potential customers by designing systems to meet their needs.

Recognize that designing from customer data is a new skill

These pioneers also knew how to see how the implications of their customer knowledge and the possibilities of technology could transform the way people work. But companies are now designing larger systems and systems that support people who are "not like us" and whose work "we do not do." That's a harder problem. Seeing how knowledge of *other* people's work should change a design is a skill and a new way of thinking for many people. Even hiring a customer into the development team doesn't guarantee the skill—most customers don't have in-depth knowledge of the technology. Just as moving from procedural to object-oriented design is a new skill requiring a new way of thinking, just as moving from forms interfaces to windowing interfaces is a new skill, moving to customer-centered, data-based design is a new skill. Coders who are new to object-oriented languages and UI designers new to windows tend to continue operating out of their old way of thinking—they create code or user interfaces that still reflect the old structures. In the same way, people who aren't used to designing from data don't find it natural to see design implications in data. Much of Contextual Design—and much of this book—is intended to help designers see design implications in customer data.

The idea that design isn't inherent in data tends to be lost when we talk of requirements *gathering* or *elicitation*. When someone checks his answering machine as soon as he walks in the office in the morning, this action says he wants to know at once who has tried to reach him. For a communications tool, this piece of data might suggest that an immediately visible notification of waiting messages is critical, perhaps by putting a blinking red light on the box. But nothing in the customer's environment declares a requirement for a blinking light. Requirements and features don't litter the landscape out at the customer site. Designers have to make this leap from fact to implication for design. And because design is implied by the customer data, what designers see in data changes based on what they are designing—a maker of office chairs would be more interested in whether the customer sat down before playing his messages. Making the shift to data-based design asks designers to learn to draw design implications out of the work, rather than implementing enhancement requests.

Don't expect to find requirements littering the landscape at the customer site

THE COMPLEXITY OF WORK

A design team reacts to their understanding of the customer by designing a solution. This isn't primarily a design for technology—how to structure and deliver a particular tool that will improve work in some way. It's the design of a new way of working that is supported by technology. We saw in the label story above that when technology is the focus for design, the work practice falls apart, and the user has to run back and forth

The complexity of work is overwhelming, so people oversimplify between computer and printer to print a label. Instead, a customer-centered design process makes work practice the focus for design. All the rest of the design elements fall out of this. But it's a difficult transformation for the engineer who is used to looking for ways to apply neat technology.

Designing work practice is a daunting task because any real work is complex and intricate. What's really involved in writing a letter? When do people decide to start fresh, and when do they start from a previous letter they wrote? How do people choose the style of a letter, and how do they maintain it throughout? What's involved in keeping an address book, choosing an address from it, and inserting it into the letter? "Writing a letter" is one simple aspect of office work, yet who really understands what's involved? But to support these customers well, designers must understand work at this level. Unless they can see and manage the complexity of real work, they can't keep it coherent for the user.

MAINTAINING A COHERENT RESPONSE

If work is to remain coherent, the system work model had better hang together. It's not good enough to get the five top issues or the three key market requests and respond to each separately. After five years of that, companies are saying to us that they no longer know what all the systems they have do or how they might fit together. To keep from losing control of the systems like this, designers have to respond to work issues not with individual features but with a *systemic response.* Such a response keeps the system work model coherent even when delivery is broken into multiple products and versions.

Furthermore, it's no longer enough to design single systems in isolation. Computer systems are now supporting so much of people's work that understanding how they fit together is critical—but because work is complex, the web of systems that supports it is also complex. Currently, most work is supported by a combination of systems so complex that no one really understands it. Consider office work again, supported by word processors, spreadsheets, financial packages, layered on operating systems extended by add-on utilities, running on hardware from several

A systemic response—not a list of features—keeps user work coherent vendors. Just try to get someone to explain to you why you can't fax from your word processor using the third-party PC-card modem you bought last week. It's too hard, even for the developers who control all the parts.

The team needs to see the work practice of its users and see the system structure as a whole. Revealing both work practice and system structure calls for a representation that makes the important issues stand out. Team members can interact over this representation, using it to present their thoughts and capture their conversation. It's this coherent representation that ensures the system hangs together, supports the customer, and gives the whole organization a single focus for parallel efforts. It's this coherent representation that continually reminds each developer how her part fits into the system as a whole and supports the overall work flow of her users. It discourages developers from focusing on one part to the exclusion of all others, inducing them to keep the system in proportion.

Diagrams of work and the system help a team think systemically

Keeping a system coherent also depends on the organization that delivers it. When development groups break the work into pieces that they can understand individually and support each piece separately, they produce a multitude of systems that don't hang together. They do this for the best of reasons: they have to ship something. Development cycles of two years and up are no longer acceptable; many teams are moving to delivering in six-month windows. The challenge is to accept this reality of the engineering world and still keep the system coherent—to recognize the overall work situation and envision the integrated solution, but deliver in small pieces that are useful on their own but can grow up into a single solution to the whole problem.

The core of any design process is supporting design thinking: the invention and development of ideas for a coherent system, based on an understanding of customers' work practice. Design thinking maintains system coherence in the face of the breadth of complexity and variety and the depth of detail in both the work and system. The challenge for Contextual Design is to support design thinking through techniques that lead to developing a coherent understanding of the work and the system's response, making both work and system concrete, explicit, and sharable and dealing with the tendency of organizational structures to pull the design apart.

THE EVOLUTION OF CONTEXTUAL DESIGN

Contextual Design grew over many years of working with design teams on different problems. As we recognized places in the process that teams had difficulty with, we modified the process or introduced new steps to address the problem. Here's a summary of how the process grew.

Contextual Inquiry (CI) was the first part of the process. Karen Holtzblatt developed it as a response to a challenge from John Whiteside: design a process that would lead to new kinds of systems rather than iterating existing systems (Holtzblatt and Jones 1995). Prototyping and usability testing could iterate an existing system, but couldn't suggest wholly new directions. CI meets the challenge by putting designers and engineers directly in the customers' work context, thereby giving them the richest possible data to invent from. From this beginning, interpersonal issues were central; cross-functional CI teams developed a shared understanding of the customer from the beginning to alleviate the transition to development. The core process for CI was worked out with Sandy Jones by working with several engineering projects.

Contextual Inquiry produces vast amounts of detail, and managing the quantity of information became difficult. So Holtzblatt adapted the affinity diagram process to reveal the order and structure in the data collected during contextual interviews. This became the classic CI process taught for many years at places such as the Conference on Computer-Human Interaction (CHI).

Paper prototyping to test a design is an adaptation of Participatory Design methods, especially those developed at Aarhus University (Ehn 1988; Greenbaum and Kyng 1991), combining them with the style of interviews used in CI. Paper prototypes were introduced to iterate designs with customers without the need to commit anything to code. In this way, a design could be tested with minimal investment.

Working intensively with design teams revealed that when designing for the customer, there wasn't a good way to represent design alternatives. The Pugh matrix process provided a way to envision several design alternatives and combine them to produce new alternatives, while keeping team members from butting heads (Pugh 1991). This process, much modified, became the visioning process in Contextual Design.

In sketching out system designs, we tended to get into unwanted arguments in the teams. UI sketches tended to divert the team into UI design prematurely, and no other formalism existed to show the structure of the system from the user's point of view. We developed the User Environment formalism as a way to capture in standard form the sketches we wanted to represent our early designs. One of the first uses of the formalism was to represent a complex design integrating nine point products, showing each product team their place in the overall design.

But we discovered that teams still had a hard time seeing design implications in the hierarchical structure of an affinity diagram. An affinity doesn't show the structure or pattern of work; it reveals issues but doesn't show how to structure a solution. Work models are a formalization of informal sketches of customer work. As we used these models in different design situations, we became more confident that they did represent the key aspects of work for most design problems, and we standardized the five that Contextual Design now includes. ➪

Finally, we found that it was too hard to lead a team through the transition from consolidated models to User Environment Design; it was still too much like magic. So we introduced an explicit visioning step to create the new design. At first we had people work out the details in redesigned sequence models, but then found they preferred to think pictorially in storyboards.

At each point in the evolution of Contextual Design, we had a process that worked well enough for the problems at hand. But at each point we recognized problems we did not have a good way to solve. Contextual Design grew by taking a problem and using our principles of design to redesign the process to address it. Solving that problem would then reveal the next, and so on, until the process got reasonably stable. The result of this evolution is the process you now have. ❑

CONTEXTUAL DESIGN

Contextual Design is a customer-centered process responding to these issues. It supports finding out how people work, so the optimal redesign of work practice can be discovered. It includes techniques that manage the interpersonal dimension of designing in cross-functional teams and keep designers focused on the data. And it leads the team through the process of discovering design implications for redesigning work practice, developing a corporate response, and structuring a system in support of the redesign.

Contextual Design provides explicit steps and deliverables for the front end of design, from initial discovery through system specification. As such it works well for organizations putting ISO 9000 or SEI-compliant processes in place: well-defined steps and measurable deliverables support the requirements of those standards for defined, repeatable processes. Though optimized for large, complex projects, Contextual Design has been successfully used on small projects as well. And because Contextual Design provides a complete structure for the front end, teams have used it very effectively as a scaffolding into which they incorporate additional techniques and processes as the need arises.

In our approach to process design, we recognize that much of what we do is to make explicit and public things that good designers do implicitly. Each of the parts of Contextual Design reflects a part of the design process that has to happen anyway—either informally in one person's head or publicly as an explicit design step. Making a step explicit makes it something that a team can do together and makes it possible to share the thinking process and results with others. It may also make the step take longer, but if you're currently going through a lot of argument in design and rework, making that step explicit may well reduce the time it takes. And if your customers are complaining about usability and integration across applications, taking the time may be what's required. Make sure, when you look at your processes, to take into account not just the formal time a step takes, but the amount of time it really takes in your organization. If your engineering team is still arguing over what functions should be included in a release two weeks before test, you are still in the requirements analysis phase no matter what your project calendar says or how much code you've written. Deciding how you will use a design process—which steps are critical, which can be omitted—is an important first step for any project. (Chapter 20 gives guidance on how to tailor Contextual Design to specific situations.)

CD externalizes good design practice for a team

The parts of Contextual Design, and the parts of the book that cover them, are as follows:

Contextual Inquiry: The first problem for design is to understand the customers: their needs, their desires, their approach to the work. Even the hacker coding in the basement has some notion of who he thinks the customers are and what they want. A customer-centered process makes an explicit step of understanding who the customers really are and how they work on a day-to-day basis. Contextual Design starts with one-on-one interviews with customers in their workplace while they work. These are followed by team interpretation sessions in which everyone can bring their unique perspective to bear on the data. This supports the team in developing a shared view of all the customers they interview. We'll describe Contextual Inquiry and how to apply it in Part 1.

Talk to the customers while they work

Work modeling: Understanding the customer is good, but customer work is complex and full of detail. At the same time it's intangible; work practice is not naturally a concrete thing to be manipulated. Designers might be able to get away without an explicit representation of a simple work domain they are familiar with. But what happens when the work domain is complex and unfamiliar? What happens when it crosses multiple departments in an organization? How do you communicate and share knowledge of a way of working? For these situations, Contextual Design provides a concrete representation. *Work models,* built during interpretation sessions, provide a concrete representation of the work of each customer interviewed. There are five kinds of work model, each providing a unique perspective on the customer. Each perspective is complete, showing the whole work practice, yet focused on a single set of issues. These work models are described in Part 2.

Represent people's work in diagrams

Consolidation: Systems are seldom designed for a single customer. But designing for a whole *customer population*—the market, department, or organization that will use the system—means being able to see the common structure inherent in the work different people do. Studying different customers will give designers a feel for the common approaches to work across the population, but it takes special techniques to make that "feel" explicit so that a team can see the common pattern without losing individual variation. The consolidation step of Contextual Design brings data from different customers together and looks across multiple customers to produce a single picture of the population a system will address. This is done through an *affinity diagram* (Brassard 1989), bringing individual points captured during interpretation sessions together into a wall-sized, hierarchical diagram showing the scope of issues in the work domain, and *consolidated work models,* showing the underlying pattern and structure of the work the design will address. Together, they show what matters in the work and guide how to structure a coherent response. Consolidation is described in Part 3.

Pull individual diagrams together to see the work of all customers

Work redesign: Any system is put in place because its designers hope to improve their customers' work practice. That improvement is often implicit, presented as the result of adopting some technological solution. In Contextual Design, the team uses the consolidated data to drive conversations about how work could be improved and what technology could be put in place to support the new work practice. The team invents improved ways to structure the work rather than focusing on technical solutions. This *vision* drives changes to the organizational structure and

Create a corporate response to the customers' issues procedures, as well as driving the system definition. Using *storyboards,* the team develops the vision into a definition of how people will work in the new system and ensures that all aspects of work captured in the work models are accounted for. (Storyboards act like stories of the future in the sense used by Rheinfrank and Evenson [1996].) This process is described in Part 4.

User Environment Design: The new system must have the appropriate function and structure to support a natural flow of work. This structure is the system work model—the new way of working implicit in the system. It's the floor plan of the new system, hidden behind user interface drawings, implemented by an object model, and responding to the customer work—but typically not made explicit in the design process. In Contextual Design, the system work model has an explicit representation in the User Environment Design. As a floor plan for the system, the User Environment Design shows the parts and how they are related to each other from the user's point of view. The User Environment Design shows each part of the system, how it supports the user's work, exactly what function is available in that part, and how it connects with other parts of the system, without tying this structure to any particular UI. With an explicit User Environment Design, a team can make sure the structure is right for the user, plan how to roll out new features in a series of releases, and manage the work of the project across engineering teams. Basing these aspects of running a project on a diagram that focuses on keeping the system coherent for the user counterbalances the other forces that would sacrifice coherence for ease of implementation or delivery. Building and using a User Environment Design for development is described in Part 5.

Structure the system work model to fit the work

Mock-up and test with customers: Testing is an important part of any systems development, and it's generally accepted that the sooner problems are found, the less it costs to fix them. Rough paper prototypes of the system design test the structure and user interface ideas before anything is committed to code. Paper prototypes support continuous iteration of the new system, keeping it true to the user and giving designers a data-based way of resolving disagreements. In prototyping sessions, users and designers redesign the mock-up together to better fit the user's work. The results

Test your ideas with users through paper prototypes of several of these sessions are used to improve the system and drive the detailed UI design. Paper prototyping is described in Part 6.

Putting into practice: In the last chapter, we look at practical issues of putting a new design process in place. You'll encounter resistance, you'll have to work with the limitations of the organization you have, and you'll have to build on the skills you have in place. Altering Contextual Design to fit your organization and your specific design problems means recognizing which parts are critical and which are less necessary in each case. What works for a two-person team won't work for a fifteen-person team; what works to design a strategy for a new market venture won't work for the next iteration of a 10-year-old system. We'll discuss common project situations, how to tailor the process to them, and how to ensure you don't lose the key features of the process along the way.

Tailor Contextual Design to your organization

Each part of the book has a similar structure: the first chapter focuses on the organizational situation and issues driving that phase of the design process. If you're looking for an overview of Contextual Design and the thinking behind it, concentrate on the first chapter of each part. The second chapter of the part describes what makes this phase of the design process customer-centered. It discusses how to make customer considerations central given the needs and constraints of this phase of design. And the third chapter of each part describes how to do the work, covering particular procedures and techniques that guide a team through the process.

This book is intended to capture our experience designing customer-centered processes to meet a wide variety of team situations and design problems. As we describe in Part 3, there's a broad commonality of work practice across any industry, so we expect the solutions we've developed to be generally useful; there will be a lot here that you can pick up and apply in your own situation. However, every situation is unique, and you should expect that you will tailor the things you pick up to your problem, team, and organization. Treat Contextual Design as a coherent design process but also as a collection of techniques and a framework for thinking. Where you have other techniques you've found to be valuable, slide them into the appropriate place in the process. This is a starting point. What you do with it is up to you.

ALAN'S STORY

I'm the project manager for a network management application. We'd done a lot of good engineering work on the application, converting it to C++ and cleaning up the architecture. We also had pretty good software development processes—we specified new features before building them instead of coding by the seat of our pants, down to the point of dealing with error conditions. But when we presented our last version to customers, they liked it but were not excited. I decided we didn't really understand our customers well enough and that we should do something about it.

Our UI designer had some experience with Contextual Design and talked me into trying it for our next version. I was told it would take 15 days, so I agreed to put four engineers on it, including myself. Then when I talked to the coach who would lead us through the project, I found out it would actually take four to six months. This was a shock. But I decided it was important, and we went ahead with three engineers and three documentation people.

We did 16 contextual interviews in all. Even during the initial interviews, I learned more about the customers' real needs. It wasn't so much that we came up with new features as that my whole understanding of the real priorities changed: things that I thought were priority 100 I realized we had to do in this release, and some cool technical features that were priority 1 moved down to 100. Throughout the project, I changed priorities of different tasks and reassigned people as I understood better what our customers needed.

We built an affinity and consolidated our models, which crystallized our understanding of customer priorities. Then we did the visioning and storyboards based on the vision. This frankly scared me; I thought if we had data from 50 or 60 people I could rely on it better. Also at this point, some of the UI designers pulled out. They were bothered by working as a team on a part of the design they used to do alone and said they didn't see the need for all this customer data. We went forward with the redesign, but then I began to get uncomfortable; some of the team didn't want to be tied to reality. They wanted to design from scratch.

I decided to cut and run. I told the team we had two weeks to develop final designs for changes to be delivered in the next release—that made everyone get very concrete. We cleaned up our ideas, and the team's architect and I wrote six specifications capturing the new design. I assigned the specifications to developers to flesh out and code on their own. Issues did come up later during development, and we would go back to the consolidated models and affinity to resolve them. Often when engineers were arguing over two alternatives, the models would suggest some third alternative that they hadn't even thought of.

We finished the release and showed it around the company. People were excited, but the real test came when we demonstrated it at our worldwide users' group meeting. For each new part of the system, I explained what we saw customers trying to do, how the new system would help them do it better, and then showed them the product actually doing it.

Our customers gave us a standing ovation. That's never happened before.

Looking back over the project, what strikes me is that we achieved this with no extra engineering effort. We didn't take longer to ship this release, or work longer hours than on any other, and we didn't have more developers. We were just better focused because we knew what was important to the customer. I'm using more of the process on my current release and am finding that's still true: the additional insight we are gaining is still worth the effort. ❑

3 Principles of Contextual Inquiry

The core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work. Do that, and you can't help but gain a better understanding of your customer.

That is the core of the technique, but we find people are generally happy to have a little more guidance. What do interviewers do at the customer's site? How do they behave? What kind of relationship allows customers to teach designers the depth of knowledge about their work necessary to design well?

In Contextual Design, we always try to build on natural human ways of interacting. It is easier to act, not out of a long list of rules, but out of a simple, familiar model of relationship. A list of rules says, "Do all these things"—you have to concentrate so much on following the rules you can't relate to the customer. It's too much to remember. A *relationship model* says, "Be like this"—stay in the appropriate relationship, and you will naturally act appropriately (Goffman 1959).

Design processes work when they build on natural human behavior

Many different models of relationship are available to us. A formal model might be scientist/subject: I am going to study you, so be helpful and answer my questions; it doesn't really matter whether you understand why I'm asking. A less formal model might be parent/child: I'll tell you what to do, and you'll do it because you want my approval (or else you'll rebel to show your independence). Each of these models brings with it a different set of attitudes and behaviors. Everyone knows what it is like

Use existing relationship models to interact with the customer when someone treats us like a child, and the resentment it generates. Ironically, the natural reaction is to behave like a child and fight back. Relationship models have two sides, and playing one side tends to pull the other person into playing the other side. Find a relationship model that is useful for gathering data, and as long as you play your role, you will pull the customer into playing theirs.

THE MASTER/APPRENTICE MODEL

The relationship between master craftsman and apprentice is an effective model for collecting data. Just as an apprentice learns a skill from a master, a design team wants to learn about its customers' work from its customers. Though the model is no longer common, it is still sufficiently familiar that people know how to act out of it. When they do, it creates the right behaviors on both sides of the relationship for learning about the customers' work. We find that people with no special background in ethnography learn how to conduct effective interviews much more quickly by acting like an apprentice than by memorizing a list of effective interviewing techniques. Building on this relationship model creates a strong basis for learning about work.

Craftsmen, like customers, are not natural teachers, and teaching is not their primary job. But they do not need to be; the master craftsman teaches while doing. A master does not teach by designing a course for apprentices to take. Nor does a master teach by going into a conference room and discussing his skill in the abstract. A master teaches by doing the work and talking about it while working. This makes imparting knowledge simple.

Teaching in the context of doing the work obviates any need for the craftsman to think in advance about the structure of the work he does. As he works, the structure implicit in the work becomes apparent because both master and apprentice are paying attention to it. It is easy for the master to pause and make an observation or for the apprentice to ask a question about something the master did. Observation interspersed with discussion requires little extra effort on the part of either master or apprentice.

When you're watching the work happen, learning is easy

Similarly, in Contextual Inquiry, team members go to the customers' workplace and observe while they are immersed in doing their work. Like the driver of a car, customers don't think about how they are working. But they can talk about their work as it unfolds. They do not have to develop a way to present it or figure out what their motives are. All they have to do is explain what they are doing, as does this user of a desktop publishing product:

> I'm entering edits from my marked-up copy here . . . I'm working in 200% magnification so I can really see how things line up. It doesn't matter that I can't see all the text in this magnification because I'm not checking for continuity or natural flow of words; I'll do that in another pass later. . . .

Even if the master were a good teacher, apprenticeship in the context of ongoing work is the most effective way to learn. People aren't aware of everything they do. Each step of doing a task reminds them of the next step; each action taken reminds them of the last time they had to take such an action and what happened then. Some actions are the result of years of experience and have subtle motivations; other actions are habit, and there is no longer a good reason for them. The best time to unravel the vital from the irrelevant and explain the difference is while in the middle of doing the work.

Seeing the work reveals what matters

This holds true for customers as well. They are not aware of everything they do or why they do it; they become aware in the doing.[1]

> Once we observed someone sorting his paper mail. He was able to tell us exactly why he saved, opened, or threw out each piece because he was in the process of making that decision.
>
> Another time, a research scientist came to the end of a painstaking series of mechanical calculations, turned to us, and said, "I guess you're surprised that I'm doing this." *He* was surprised at how inefficient he was, once he stopped to think about it.

But it is not natural to stop your work to think about it; the apprentice relationship provides the opportunity to do so.

Talking about work while doing it allows a master craftsman to reveal all the details of a craft. As he works, he can describe exactly what he is doing and

Seeing the work reveals details

[1] Polanyi (1958) discusses what tacit knowledge people have available for discussion at different times.

why. When either master or apprentice observes a pattern or principle in action, he can point it out immediately.

Customers who describe what they are doing while doing it, or talk about a prior event while in their work, have the same kind of detail available to them. Every action they take and every object around them helps them talk about the details of their work.

> One customer said he would not use a manual's index to find the solution to a problem: "It's never in the index." He could not say what led him to this conclusion, what he had looked up and failed to find. All his bad experiences were rolled up into one simple abstraction: it's not there. But when we watched him looking things up, we could see that he was using terms from his work domain, but the index listed parts of the system. We learned what the problem was and what we could do to fix it.

People sometimes don't even remember how to do their jobs themselves; instead, they depend on the environment and things in it to tell them what to do:

> A customer was unable to describe how she made her monthly report. When asked to create it, she pulled out her last report and started filling in the parts. The old report was her reminder of how to produce the next one.

Talking about work while doing it protects the master craftsman and the customer from the human propensity to talk in generalizations that omit the detail designers need. When the work's right there, the details, even details people do not normally pay attention to, are available for study and inquiry.

The apprentice learns the strategies and techniques of a craft by observing multiple instances of a task and forming his own understanding of how to do it himself. This understanding incorporates the variations needed to do the task well under a variety of circumstances. The master craftsman can communicate techniques and strategies without articulating them. By watching instance after instance, the apprentice builds up a big picture of how to do the work.

Seeing the work reveals structure

In the same way, interviewers observing multiple events and multiple customers learn to see the common strategies underlying the work. Once they understand the basic strategies, they can start to imagine a system that would support those strategies. For example, a basic pattern in coding is work on the code, test it, and see the results. Identifying bugs to fix leads back to working on the code. But this pattern holds true not only for code, but for creating analysis and design models and automated tests as well. We uncovered this pattern by observing multiple people working on multiple systems of varying complexity. We could then structure the CASE system we were designing to facilitate movement through this cycle. (Part 3 discusses making common patterns and strategies explicit.)

Every event serves as the starting point for discussing similar events in the past. In this way apprentices learn from experience gained by a master before their apprenticeship started. A particular occurrence or task reminds the master of other interesting times this event or task happened. If the event is reasonably close in time, the story is concrete and detailed. It is the retelling of a particular event, told while the master is immersed in doing the same activity with all the triggers and reminders doing that activity provides.[2]

Every current activity recalls past instances

A design team typically has less time to spend with its customers than the years needed for an apprenticeship. But in the same way that an apprentice can learn from the master's experience, interviewers can learn about events that occurred in the past. Events that occur while the interviewer is present remind customers to talk about events that happened previously. The artifacts of work—papers, forms, notes, clipboards, and so forth—trigger conversations about how they were used, how they were created, and how their structure supported their use in a particular instance.

> A customer describing how she learned a feature told us, "I looked it up in the documentation." But when we asked her to look it up again, she was able to show us: "I looked the function up in the index and scanned the section. I saw this icon in the margin that I recognized from the screen, so I read just this paragraph next to it. It told me all I needed to know." The documentation provided the context she needed to recover a detailed story, and the detail revealed aspects that had been overlooked—that the icon was her visual cue to the relevant part of the page.

[2] Orr (1986) describes such storytelling to transmit knowledge among modern-day system managers for similar reasons.

Contextual Inquiry seeks to provide rich detail about customers by taking team members into the field. Once there, apprenticeship suggests an attitude of inquiry and learning. It recognizes that the customer is the expert in their work and the interviewer is not. An interviewer taking on the role of apprentice automatically adopts the humility, inquisitiveness, and attention to detail needed to collect good data. The apprentice role discourages the interviewer from asking questions in the abstract and focuses them on ongoing work. And customers can shape the interviewer's understanding of how to support their work from the beginning, without having to prepare a formal description of how they work or what they need.

Contextual Inquiry is apprenticeship compressed in time

The Four Principles of Contextual Inquiry

Apprenticeship is a good starting point, but it is only a starting point. Unlike apprentices, interviewers are not learning about work in order to do it; they are learning about it in order to support it with technology. Interviewers cannot afford to spend the time an apprentice would take to learn the work. Unlike an apprentice, members of the design team contribute their own special knowledge about technology and what it can do. Apprentices learn a single job, but different projects may require the team to study a widely varying work practice—from the surgeon in the operating theater, to the manager in a high-level meeting, to the secretary at a desk, to the family in front of the video game. Designers meet the needs of a whole market or department, so they must learn from many people—individuals doing the same kind of work and individuals doing very different tasks and taking on different roles in order to get the work done.

Contextual Inquiry tailors apprenticeship to the needs of design teams

The basic apprenticeship model needs modifications to handle a design team's needs and situation. Four principles guide the adoption and adaptation of the technique: *context, partnership, interpretation,* and *focus.* Each principle defines an aspect of the interaction. Together, they allow the basic apprenticeship model to be molded to the particular needs of a design problem. We will describe each principle and how to use it in turn.

CONTEXT

The principle of *context* tells us to go to the customer's workplace and see the work as it unfolds (Whiteside and Wixon 1988). This is the first and most basic requirement of Contextual Inquiry. Apprenticeship is a fine example of doing this; the apprentice is right there to see the work. All the richness of real life is there, able to jog the customer's memory and available for study and inquiry. The customer made a phone call in the middle of doing a task. Is this relevant to the work? Was she calling on an informal network of experts to get help in the task? Someone stops by to get a signature on a form. What is the customer's role in this approval process? Do they talk about it before she signs? What are the issues?

Go where the work is to get the best data

Context tells us to get as close as possible to the ideal situation of being physically present. Staying in context enables us to gather *ongoing experience* rather than *summary experience,* and *concrete data* rather than *abstract data.* We'll describe each of these distinctions in turn.

SUMMARY VS. ONGOING EXPERIENCE. We are taught from an early age to summarize. If someone asks a friend about a movie she saw last week, she does not recount the entire plot. She gives overall impressions, one or two highlights, and the thing that most impressed or disgusted her. (Never ask a seven-year-old that question—they haven't yet learned to summarize and *will* tell you the entire plot of the movie in excruciating detail.) Ask people to tell you about their experience with a new system, and they will behave just the same way. They will give their overall impressions and mention one or two things that were especially good or bad. They will have a very hard time saying exactly why the good things were important, or why the bad things got in the way. That would require that they be able to talk about the details of their work, which is very hard to do.

We once asked a secretary how she started her day. Her answer was, "I guess I just come in and check my messages and get started." She wasn't able to go beyond this brief summary overview. It was the first thing in the morning and she had just arrived at the office, so we asked her to go ahead and do as she would any other morning. She unhesitatingly started her morning routine, telling us about it as she went: "First I hang up my coat, then I start my computer. Actually, even before that I'll see if my boss has left something on my chair. If he has, that's first priority. While the computer's coming up, I check the answering machine for urgent messages. There aren't any. Then I look to see if there's a fax that has to be handled right away. Nope, none today. If there were, I'd take it right in and put it on the desk of whoever was responsible. Then I go in the back room and start coffee. Now I'll check the counters on the copier and postage meter. I'm only doing that because today's the first of the month. . . ."

This person's morning routine has a definite structure: first she checks all her communication mechanisms to see if there is an immediate action that needs to be taken, then she starts the regular maintenance tasks of the office. But this structure is invisible to her. It would not even occur to most people as a topic of conversation.

Avoid summary data by watching the work unfold

The job of the interviewer is to recognize work structure. Discovery of work structure arises out of this level of detail about mundane work actions. Summary experience glosses over and hides this detail. Being present while the work is ongoing makes the detail available.

ABSTRACT VS. CONCRETE DATA. Humans love to abstract. It's much easier to lump a dozen similar events together than to get all the details of one specific instance really right. Because an abstraction groups similar events, it glosses over all the detail that makes an event unique. And since a system is built for many users, it already needs to abstract across all their experience. If designers start from abstractions, not real experience, and then abstract again to go across all customers, there is little chance the system will actually be useful to real people. Even in the workplace, customers easily slide into talking about their work in the abstract. But there are signals that indicate the customer needs to be brought back to real life.

If the customer is leaning back and looking at the ceiling, he is almost always talking in the abstract. This is the position of someone who will not allow the reality all around him from disrupting the conception he is building in his brain. Someone talking about real experience leans forward, either working or pointing at some representation of what he is talking about. Words indicating the customer is generalizing are another signal. If the customer says, "generally," "we usually," "in our company," he is presenting an abstraction. Any statement in the present tense is usually an abstraction. "In our group we do . . ." introduces an abstraction; "that time we did . . ." introduces real experience.

The best cure is to pull the customer back to real experience constantly. Every time you do this, you reinforce that concrete data matters, and you make it easier to get concrete data next time. If the customer says, "We usually get reports by email," ask, "Do you have one? May I see it?" Use the real artifacts to ground the customer in specific instances. If the customer says, "I usually start the day by reading mail," ask, "What are you going to do this morning? Can you start?" Return the customer to the work in front of him whenever possible.

Avoid abstractions by returning to real artifacts and events

Sometimes the work that you are interested in happened in the past and you want to find out about it, so you need to elicit a *retrospective account.* Retelling a past event is hard because so much of the context has been lost. People are prone to giving a summary of a past event that omits necessary detail. Most people will start telling a story in the middle, skipping over what went before. They will skip whole steps as they tell the story. The interviewer's job is to listen for what the customer is leaving out and to ask questions that fill in the holes. Here is an example of walking a customer through a retrospective account. The customer is talking about how they dealt with a report. We've interpolated the dialog with the missing steps that the interviewer is hearing in the data.

Span time by replaying past events in detail

Customer: *When I got this problem report I gave it to Word Processing to enter online—*

(Why did she decide to give it to Word Processing? Did she do anything first?)

Interviewer: *So you just handed it on automatically as soon as you got it?*

C: *No, it was high priority, so I read it and decided to send a copy to the Claims department.*

(How did she decide it was high priority? Is it her decision?)

I: *How did you know it was high priority?*

C: *It has this green sticker on it.*

(Someone else made the decision before the report ever got here. Who and when?)

I: *Who put on the green sticker?*

C: *That's put on by the reporting agency. They make the decision about whether it's high priority and mark the report.*

(We can better pursue how the reporting agency makes the decision with them; we'll only get secondhand information from this user. Instead of trying to go further backward, look for the next missing step forward: doesn't Claims get a more personal communication than just the report?)

I: *Did you just send it on to Claims, or did you write them a note about why they needed to see it?*

C: *Oh, I always call Claims whenever I send them one of these reports.*

At each step, the interviewer listened for steps that probably happened but the customer skipped and then backed the customer up to find out. In this process, the customer walked through the steps in her mind, using any available artifacts to stimulate memory, and recalled more about the actual work than she would if allowed to simply tell the story in order. Using retrospective accounts, the interviewer can recover past events and can also learn more about events in progress. If the end of a story hasn't yet happened, the most reliable way to learn about that kind of situation is to go back to a previous occurrence that did complete and walk through it. Trying to go forward and find out what will happen next forces the customer to make something up; going to another past instance allows the customer to stay concrete.

The key to getting good data is to go where the work is happening and observe it while it happens. Observing ongoing work keeps the customer concrete and keeps them from summarizing. Keeping to the apprenticeship model helps with this; the apprentice wants to see and assist with real work. If the customer starts telling stories, the interviewer can (exerting a little more control than an actual apprentice would) either redirect him to ongoing work or delve into the story, using a retrospective account to get all the detail possible.

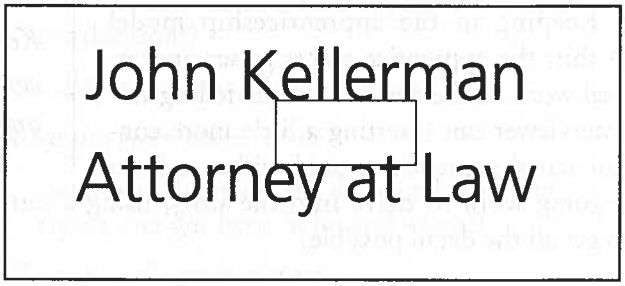
Keep the customer concrete by exploring ongoing work

Partnership

The goal of *partnership* is to make you and the customer collaborators in understanding his work. The only person who really knows everything about his work is the one doing it. The traditional interviewing relationship model tilts power too much toward the interviewer. The interviewer controls what is asked, what is discussed, and how long is spent on a topic. This won't get you design data—you don't know what's important to pay attention to, and you don't know what will turn out to matter. The apprenticeship model tilts power, if anything, too much toward the master-customer. It suggests that the customer is in full control, determining what to do and talk about throughout the interview. Traditional apprenticeship would reduce the interviewer to asking a few questions for clarification, at best.

Help customers articulate their work experience

This is too limiting for an interviewer understanding work practice. An interviewer's motive in observing work is not that of the apprentice. Apprentices want to know how to do the work; interviewers want data to feed invention of a system that supports the work. Apprentices are assumed to bring no useful skills to the relationship. Any skills they happen to have they subordinate to learning the way the master goes about working. Designers may not be experts in doing the work, but they must develop expertise in seeing work structure, in seeing patterns and distinctions in the way people organize work. An interviewer has to create something that looks more like a partnership than like an ordinary apprenticeship. This allows them to engage the customer in a conversation about the work, making the customer aware of aspects of the work that were formerly invisible and bringing the customer into a partnership of inquiry into the work practice.



In one interview with a user of page layout software, the user was positioning text on the page, entering the text and moving it around. Then he created a box around a line of text, moved it down until the top of the box butted the bottom of the line of text, and moved another line of text up until it butted the bottom of the box. Then he deleted the box.

Interviewer: *Could I see that again?*

Customer: *What?*

I: *What you just did with the box.*

C: *Oh, I'm just using it to position this text here. The box doesn't matter.*

I: *But why are you using a box?*

C: *See, I want the white space to be exactly the same height as a line of text. So I draw the box to get the height.* (He repeats the actions to illustrate, going more slowly.) *Then I drag it down, and it shows where the next line of text should go.*

I: *Why do you want to get the spacing exact?*

C: *It's to make the appearance of the page more even. You want all the lines to have some regular relationship to the other things on the page. It's always hard to know if it really makes any difference. You just hope the overall appearance will be cleaner if you get things like this right.*

I: *It's like everything you put on the page defines a whole web of appropriate places for the other things to go.*

C: *That's right. Everything affects everything else. You can't reposition just one thing.*

This is a common pattern of interaction during an interview. While work is progressing, the customer is engrossed in doing it, and the interviewer is busy watching the detail as it unfolds, looking for pattern and structure, and thinking about the reasons behind the customer's actions. At some point the interviewer sees something that doesn't fit, or notices the structure underlying an aspect of the work, and interrupts to talk about it. This causes a break in the work, and both customer and interviewer *withdraw* from doing the work to discuss the structure that the interviewer found. It is as though they stepped into a separate conceptual room. The customer, interrupted in the moment of taking an action, can say what he is doing and why. The interviewer, looking at work from the outside, can point out aspects the customer might take for granted. By paying attention to the details and structure of work, the interviewer teaches the customer to attend to them also. When the conversation about structure is over, the customer *returns* to ongoing work, and the interviewer returns to watching. This *withdrawal and return* is a basic pattern of Contextual Inquiry: periods of watching work unfold, interspersed with discussions of how work is structured.

Alternate between watching and probing

Over the course of an interview, customers become sensitized to their own work and how it could be improved. Questions about work structure reveal that structure to them so they can start thinking about it themselves. "It's like everything you put on the page defines a whole web of appropriate places for the other things to go." This comment suggests a way of thinking about the work. It makes a previously implicit strategy explicit and invites a conversation about that strategy. Soon customers start interrupting themselves to reveal aspects of work that might otherwise have been missed. Over the course of the interview, a true partnership develops, in which both customer and interviewer are watching work structure, and in which both are thinking about design possibilities. (See Chin et al. [1997] on making customers participants in analyzing their own work.)

Teach the customer how to see work by probing work structure

Members of a design team also have special knowledge about how to use technology. They notice problems that they can solve and allow them to distract them from the work. They naturally figure out a solution to any problem or apparent problem that presents itself. But this is a distraction from the interview because, rather than listening to whatever the customer is saying, the interviewer is off thinking about the great thing she could make. She can't pay attention to the work while designing something in her mind.

It's not useful to tell designers not to design in the moment—they will anyway. One of the principles of Contextual Design is to work with people's propensities wherever possible. So rather than forbid designing in the moment, we manage it by allowing the interviewer to introduce her idea immediately. The customer is in the middle of doing the work that the idea is intended to support. There is no better time to get feedback on whether the idea works. If the idea works, the interviewer understands the work practice and has a potential solution. If the idea fails, the interviewer did not really understand what mattered in the work. By sharing the idea, the interviewer improves her understanding of the work and checks out her design idea at the same time. In addition, the idea suggests to the customer what technology could do. Customers start to see how technology might be applied to their problem.

Find the work issues behind design ideas

Articulating work structure and correcting design ideas during the interview gives the customer the power to shape the way designers think about the work. Any iterative technique (such as rapid prototyping or Participatory Design) enables customers to shape a proposed design. But iterating an existing design can only make small modifications to its structure. That initial structure—the first prototype—was driven by whatever way of thinking about the work that the designer had when she started. A process is truly customer-centered when customers can change designers' initial understanding of the work. Sharing interviewers' initial, unformed ideas with the customer and articulating work practice together allows customers to alter the team's initial thinking, opening the possibility of radical changes in system purpose and structure.

Let the customer shape your understanding of the work

Avoiding other relationship models. The danger in all of this is that customer or interviewer will fall back into more familiar models of relationship. There are many other models available, each with its own set of problems. If you fall into one of these models during an interview, you will pull the customer into the other side of the relationship, prompting behavior that gets in the way of gathering data. If you are aware of what these other relationships are like, you can notice when you fall into them and take actions to shift back into the right relationship. Here are some common pitfalls:

Interviewer/interviewee: Interviewer and customer start to act as though there were a questionnaire to be filled out. You ask a question, which the customer answers and then falls silent. You, anxious that the interview go well, ask another question, which the customer answers and then falls silent again. The questions are not related to ongoing work because ongoing work has ceased. The best solution for this is to suggest returning to ongoing work, which effectively prevents this question/answer interaction.

You aren't there to get a list of questions answered

Expert/novice: As a representative of the design team, you go in with the aura of the expert. You are the one designing the system, with all the technical knowledge. You have to work to get the customer to treat you as an apprentice. The temptation of taking the expert role back is always present, especially when the customer is trying to use a system that you developed. Set the customer's expectations correctly at the beginning by explaining that you are there to hear about and see their work because only they know their own work practice. You aren't there to help them with problems or answer questions. Then, should the customer ask for help (or should you forget and volunteer help), step out of the expert role explicitly: "I'll never understand the problems with our system if I spend the whole time helping you. Why don't you go ahead and do what you would do if I weren't here, and at the end I'll answer any questions that remain." The only exception to this rule is if the customer is so stuck that he will not be able to do any more of the work you came to see. In that case, give enough information to help him find his way out of the problem. Then you'll have to say all

You aren't there to answer questions either over again that you came to see how he does things and he shouldn't depend on you for answers.

Guest/host: Because it is the customer's workplace and the customer is a stranger, it is easy to act like a guest. A guest is polite and not too nosy. A host is considerate and tries to make the guest comfortable by seeing to his needs. Unfortunately, none of this has much to do with doing real work. If you find yourself feeling like a guest, move quickly past the formal relationship to the role of partner in inquiry. This is where sensitivity to culture matters. If the customer won't be comfortable until you've had a cup of coffee, then have it and move into doing work. The relationship should feel like the kind of intimacy people strike up on airplanes, when they tell things that they would not ordinarily share with a stranger. Here, intimacy doesn't come from personal talk; it comes from a shared focus on the work. Move closer. Ask questions. Be nosy. Ask to see anything the customer touches, and get them to tell you about it. You will know you created the relationship you want when the customer says to you, "Come over here—you want to see this." The more you get them to tell you about themselves, the more you will move out of the formal role.

It's a goal to be nosy

Partnership transforms the apprenticeship relationship into a mutual relationship of shared inquiry and discovery of the customer's work. It retains the close working relationship from apprenticeship while equalizing the power imbalance. This results in an intimate relationship that allows for inquisitiveness about the details of the work. The relationship is maintained by honesty and openness on the part of the interviewer, who reveals insights and ideas as they occur, and guards against allowing inappropriate relationship models that take the conversation off topic and prevent getting good data.

Partnership creates a sense of a shared quest

INTERPRETATION

It is not enough only to observe and bring back observations. Interpretation is the assignment of meaning to the observation—what it implies about work structure and about possible supporting systems. The language our field uses to describe gathering data for design—data *gathering,* field *research,* requirements *elicitation*—

Determine what customer words and actions mean together suggests that what matters is the facts about the work. Good products, by implication, are based on facts. Interpretation says that good facts are only the starting point. Designs are built on the interpretation of facts, on what the designers claim the facts mean. Here's an illustration:

> In working with one user of an accounting package, we learned that she kept a sheet of accounts and account numbers next to her screen. Here are some interpretations of what this fact might mean and what it might imply for our design:
>
> 1. Perhaps account numbers are necessary but hard to remember, and all we need to do is make the cross-reference easier. We could put the cross-reference between numbers and names online.
> 2. Perhaps numbers are unnecessary, a holdover from paper accounting systems, and all that is needed is a way to refer to an account uniquely. We could get rid of account numbers altogether and identify them only by name.
> 3. Perhaps compatibility with paper systems is necessary, but referring to accounts by name is more convenient. We could keep the numbers but allow names to be used anywhere numbers are used.

Which of these designs is best? It depends on which interpretation is correct; the fact alone does not allow us to choose. The designer must choose which interpretation to lay on the fact. It's the interpretation that drives the design decision.

Interpretation is the chain of reasoning that turns a fact into an action relevant to the designer's intent. From the *fact,* the observable event, the designer makes a *hypothesis,* an initial interpretation about what the fact means or the intent behind the fact. This hypothesis has an *implication* for the design, which can be realized as a particular *design idea* for the system. For example, the second interpretation above starts with the fact (the chart of accounts is kept next to the screen) and makes the hypothesis that this is just a holdover from paper accounting systems. This interpretation, if true, has implications for the system: it doesn't matter whether the system provides numbers, but it must provide some way to refer to an account unambiguously. This implication can be acted on by requiring

Design ideas are the end product of a chain of reasoning the system to identify accounts through unambiguous names only. This entire chain of reasoning happens implicitly any time anyone suggests a design idea. Usually it happens so fast, only the final idea is made explicit. But the whole chain must be valid for the design idea to work.

If the data that matters is the interpretation, we must have a way to ensure it is correct, and we can only do that by sharing it with the customer. We fail in the entire purpose of working with customers if we do not share and validate our interpretations of their work—the most important data we bring back would not be validated. Sharing interpretations ensures that the work is understood correctly. Sharing design ideas walks the chain backwards; if the idea doesn't fit, some link in the chain was wrong. When it's the customer coming to you with design ideas in the form of wish lists, treat them the same way: walk the chain backwards to understand the work context driving the wish. Understanding the underlying work practice yields much more flexibility in how to respond—many design ideas can spring from a single origin. Understanding and fixing the underlying problem in the work practice can address many design ideas with a single solution. The partnership we have built up with the customer provides a natural context for sharing observations of structure and interpretations of their meaning.

Design is built upon interpretation of facts—so the interpretation had better be right

Can you really check an interpretation just by sharing it with the customer, or will that bias the data? Will customers be prone to agree with whatever you say? In fact, it is quite hard to get people in the middle of doing work to agree with a wrong interpretation. It's not at all hypothetical for them because they are in the midst of the work. The statement that doesn't fit is like an itch, and they poke and fidget with it until they've rephrased it so it represents their thought well:

Sharing interpretations with customers won't bias the data

> "It's like a traveling office," you say, looking at how a salesman has set up his car. "Well—like a traveling *desk,*" he responds.

The difference between the two is small but real, and people will be uncomfortable until they get a phrasing that fits exactly.

Furthermore, remember that the data that matters is the interpretation of the facts, not the facts themselves. You can't form an interpretation without getting involved with the events, without trying to make sense of them *for you.* Where an event contradicts your assumptions, you have to inquire and probe, or you'll never be able to replace your current, flawed understanding with one that works. This probing is driven by your expectations and prejudices, yet it is the only way your prejudices can be overturned.

Finally, since customers are not generally experts in seeing the structure of their own work, the interpretation you suggest shows them what to pay attention to. Open-ended questions give the customer less guidance in thinking about their work than an interpretation and result in less insight.

Sharing interpretations teaches customers to see structure in the work

> We might have asked a customer who was starting her workday, "Do you have a strategy for starting the day?" Even though the customer just went through the morning routine, she is not used to thinking about strategy driving ordinary work events. The most likely response would be "No, not particularly"—or a blank stare. But if asked, "You check for any urgent communication first, no matter what form it might have come in?" she can compare this statement of strategy to her own experience and validate it or refine it. She might respond, "Yes, lots of things here are time-critical and we have to deal with them right away"—simply validating the interpretation, adding detail but leaving it essentially unchanged. In fact, she responded, "Actually, things from my boss are most important because they are for me to do. Messages on the answering machine or faxes might be for anyone"—refining the interpretation, accepting the broad outline, but adding a new distinction.

Because customers respond to the interpretation in the moment of doing the work, they can fine-tune it quite precisely. Customers commonly make slight changes in emphasis such as those above to make the interpretation exact. They can do this because they are given a starting point that they can compare with the experience they are now having and adjust it, rather than having to start from scratch. In this way, we use the close relationship

Customers fine-tune interpretations

between interviewer and customer to get very reliable data. In fact, it's the only way to get reliable data; if we don't check it with the customer immediately, we take away an understanding that is at least partially made up.

However, interviewers do need to be committed to hearing what the customer is really saying. They may say "no" to an interpretation, but to be polite may not say "no" directly. Here are some indirect ways customers say "no."

"Huh?"—This means the interpretation was so far off that it had no apparent connection to what the customer thought was going on.

"Umm . . . could be"—This means "no." If the interpretation is close, the customer will nearly always respond immediately. A pause for thought means that they are trying to make it fit their experience and cannot.

"Yes, but . . ." or "Yes, and . . ."—Listen carefully to what follows the "but" or "and." If it is a new thought, this is the right interpretation and yours was wrong. If it builds on yours, this is a confirmation with a twist or with additional information. Customers say "yes" by twinkling their eyes at you as they realize your words match their experience or by elaborating on what you said—or by saying "yes" flatly, as if the whole point was obvious.

We ensure the interpretation is true by creating and maintaining the right relationship with our customer. With apprenticeship as the starting point, we create a close, intimate partnership. Partnership is a natural consequence of a contextual interview. For the entire time, we pay close attention to this person, what he does and how he does it, what gets in his way, and everything that's important to him. We take an interest. Most people have never been the focus of so much positive attention or had such an extended opportunity to talk about what they do. They become invested in making sure we get it right—that we see everything that's relevant and that we take away the exact right shade of meaning. The closer our relationship and more invested the customer, the less willing they are to allow us to leave thinking the wrong thing. This is our safeguard that our understanding is true to their experience.

Nonverbal cues confirm interpretations

Focus

Focus defines the point of view an interviewer takes while studying work. Once the interviewer is in the customer's workplace and has created a collaborative relationship with her, what should he pay attention to? What aspects of work matter and what don't? If the customer has control over what matters, how can the interviewer steer the conversation at all? The apprentice learns whatever the master knows, and the master decides what's important. But the interviewer needs data about a specific kind of work. The interviewer needs to guide the customer in talking about the part of her work relevant to the design. *Focus* gives the interviewer a way to keep the conversation on topics that are useful without taking control entirely back from the customer. Focus steers the interview the same way that friends steer conversations with each other. The topics the friends care about—the topics in their focus—are what they spend time on. Anything one friend raises that the other doesn't care about is allowed to drop without discussion.

Clear focus steers the conversation

Taking a focus is unavoidable. Everyone has an entering focus, a whole life history defining what they notice and what they don't. Consider three interviewers watching a scientist go about her work:

> One interviewer, a software developer, notices the quantities of paperwork the scientist uses to define the procedure she follows, to record her actions, and to report her results.
>
> Another interviewer is more familiar with the lab technology and sees the kind of instruments she has and the problems she has getting them set up and calibrated.
>
> The third interviewer was once a scientist and sees how the scientist moves about her lab, getting out glassware and chemicals and putting them on the bench near the equipment she will use.

Each interviewer sees a different aspect of the work, all of which are "true," but which may be more or less relevant, depending on what is being designed.

Having a focus means that the interviewer sees more. The interviewer who knows that paperwork is important will learn to distinguish the different kinds of paperwork: the method that defines what the scientist will do, the notebook that records her actions for her experiment, the log books that record calibrations of equipment for the lab, and the formal report of her results. Each of these distinctions serves as the starting point for a new inquiry, pushing the interviewer's understanding of the lab work wider and wider. A focus gives the interviewer a framework for making sense of work.

Focus reveals detail

To ensure the team sees aspects of work important to the problem at hand, we set focus deliberately to guide the interview toward relevant aspects of work. This *project focus* gives the team a shared starting point, which is augmented by each person's entering focus so they each bring their unique perspective to bear. (We discuss how to set focus for different types of problems in the next chapter.)

If focus reveals detail within the area it covers, it conceals aspects of work that it does not cover. Different people will naturally see different things. Someone who notices paperwork cannot help but notice when papers are being dragged around the lab; someone who never thought about paperwork cannot help but overlook it until his attention is drawn to it. Meanwhile the first interviewer is ignoring physical movement around the lab to get equipment, to the next lab to borrow supplies that have run short, and into another scientist's office to consult on the method used. These aspects of work may be equally important to the design problem. The first interviewer's focus has revealed rich detail in the use of paper, but how can she expand her focus and learn about the other aspects of work? First, we set focus deliberately to give the team a common starting point, an initial way to see the work, allowing them to build their own distinctions and interpretations on that base. Then, we use group interpretation in the cross-functional team to allow team members to learn and take on each other's focus over time and bring their own focus to bear on each other's interviews (we discuss these sessions in Chapter 7). Finally, during the interview, we use *intrapersonal triggers*—the interviewer's own feelings—to alert the interviewer when they are missing something.

Focus conceals the unexpected

How to expand focus. Pay attention to intrapersonal triggers to create a deliberate paradigm shift, from the understanding of the work the interviewer started with to the understanding of work that is real for the customer interviewed and relevant to the design concern. The interviewer must be committed to seeing where an understanding does not fit and changing it, not to confirming existing expectations. Inner triggers are flags telling the interviewer when an opportunity for breaking a paradigm and expanding the entering focus exists. They work because your own feelings tell you what is happening in the interview and how to act to fix it. Here are some triggers to watch out for:

Internal feelings guide how to interview

Surprises and contradictions: The customer says something, or you see them do something, that you know is "wrong." It's something no one else would do, something totally idiosyncratic. Or else it's just random; they had no particular reason for doing it. Any one of these reactions is a danger signal. It means that you are—right now—allowing your preexisting assumptions to override what the customer is telling or showing you. The tendency is to let it pass as irrelevant; the solution is to do the opposite. Take the attitude that nothing any person does is done for no reason; if you think it's for no reason, you don't yet understand the point of view from which it makes sense. Take the attitude that nothing any person does is unique to them; it always represents an important class of customers whose needs will not be met if you don't figure out what's going on. Act like the apprentice, who always assumes a seemingly pointless action hides a key secret of the trade. Probe the thing that is unexpected and see what you find.

Nods: The customer says something that fits exactly with your assumptions, and you nod. This is the reverse of the first trigger, and it is tricky. What you are doing when you nod is saying that you can hear the customer's words, match them with your own experience, and know as a result that everything that happened to you happened to them. Is this a safe assumption? Instead, take the attitude that everything is new, as if you had never seen it before. The apprentice never assumes the master has no more to teach. Do they *really* do that? Why would they do that? What's motivating them? Look for the paradigm shift. Look for ways that what they are doing differs from what you expect.

What you don't know: The customer says something technical that you just didn't understand or is explaining something and you just aren't getting it. Now what? Are you going to admit your ignorance? Wouldn't it be easier to research the subject a bit back at the office? No, admit your ignorance. Make the customer go back and take the explanation step-by-step. Treat this as a good opportunity to step away from the expert role. You are there to learn, and you might as well learn about the technology, too. No one else will be able to tell you better what this individual is talking about. Even if the customer doesn't really understand it either, the extent of their knowledge and misinformation can be valuable for design. Furthermore, if you don't ask, you'll get more and more lost as the conversation continues.

The easiest way to design a system is from your own assumptions and prejudices. Breaking out of your preconceived notions of what the system should be and how it should work is one of your hardest design tasks. Using the customer to break your paradigm intentionally counterbalances the natural propensity to design from assumptions. Triggers alert you to specific opportunities during the interview to widen your entering focus, and the open dialog encouraged by apprenticeship allows you to inquire when you need to.

Commit to challenging your assumptions, not validating them

THE CONTEXTUAL INTERVIEW STRUCTURE

The principles of Contextual Inquiry guide the design of a data-gathering situation appropriate to the problem at hand. The principles say what needs to happen to get good data, but the design problem and the nature of the work being studied control the exact procedure to use. Studies of office work can be conducted much more simply than studies of surgical procedures. The most common structure for Contextual Inquiry is a contextual interview: a one-on-one interaction lasting two to three hours, in which the customer does her own work and discusses it with the interviewer. Each interview has its own rhythm, set by the work and the customer. But they all share a structure that helps interviewer and customer get through the time without losing track of what they are supposed to do. Every interview has four parts:

The conventional interview: You, as the interviewer, and the customer need to get used to each other as people. Running the first part of the interview as a conventional interaction helps with that. You introduce yourself and your focus, so the customer knows from the outset what you care about and can start with work relevant to the focus. You promise confidentiality, get permission to tape, and start the tape recorder. Explain that the customer and her work is primary and that you depend on the customer to teach you the work and correct your misunderstandings. You ask for any opinions about the tools the customer uses (if relevant) and get an overview of the job and the work to be done that day. This is summary data, not contextual data, so don't pursue any issues; instead, watch to see if they come up in the body of the interview and pursue them then, when they are in context. Unless the work domain is unfamiliar, this part should last no more than 15 minutes.

Get to know customers and their issues

The transition: The interviewer states the new rules for the contextual interview—the customer will do her work while you watch, you will interrupt whenever you see something interesting, and the customer can tell you to hold off if it's a bad time to be interrupted. Anytime you want to break social norms, it's best to define the new rules for social interaction so everyone knows how to behave appropriately. If you declare "lady's choice," ladies will ask men to dance and no one feels awkward. Here, you want to create the new rules for the contextual interview, so you state them explicitly. This should take all of 30 seconds, but it's a crucial 30 seconds; if you don't do it explicitly, you run the risk of spending the entire time in a conventional interview.

Explain the new rules of a contextual interview

The contextual interview proper: The customer starts doing her work task, and you observe and interpret. This is the bulk of the interview. You are the apprentice, observing, asking questions, suggesting interpretations of behavior. You are analyzing artifacts and eliciting retrospective accounts. You are keeping the customer concrete, getting back to real instances and drawing on paper when the customer draws in the air to describe something she doesn't have in front of her. You are taking copious notes by hand the whole time; don't depend on the tape to catch everything. You are nosy—after a phone conversation, you ask what it was about. Follow her around—if she goes to the files, you go along and peer over her shoulder. If she goes down the hall, you tag along. If someone comes to the door and looks diffident about interrupting, you tell him to come on in. And, of course, if the customer says she needs a break, you let her

Observe and probe ongoing work

have one. The principles of context, partnership, interpretation, and focus guide your interaction during the interview.

The wrap-up: At the end of the interview, you have a chance to wrap up your understanding of the work she does and her position in the organization. Skim back over your notes and summarize what you learned, trying not to repeat verbatim what happened, but saying what is important about the work, to her and to the organization. This is the customer's last chance to correct and elaborate on your understanding, and she usually will. Allow 15 minutes for the wrap-up.

Feed back a comprehensive interpretation

Running a good interview is less about following specific rules than it is about being a certain kind of person for the duration of the interview. The apprentice model is a good starting point for how to behave. Then the four principles of Contextual Inquiry modify the behavior to better get design data: *context,* go where the work is and watch it happen; *partnership,* talk about the work while it happens; *interpretation,* find the meaning behind the customer's words and actions; and *focus,* challenge your entering assumptions. If all these concepts start to become overwhelming, go back up to the higher-level idea of apprenticeship. You want the attitude of an apprentice; you want to create an intimate relationship in which you and the customer collaborate in understanding their work, using your focus to help determine what's relevant. That's enough to run a good interview.

Contextual Inquiry in Practice 4

What are we supposed to do?" an engineer asked us. "Knock on people's doors, asking them to let us watch them use our product?" The answer in this case was "Yes, do that." Not without setting up the visit ahead of time, of course, and there's some planning to do, but in the end it all comes down to showing up and watching. Sometimes the most difficult barrier to introducing a new way of working is people's assumptions about what is or is not "done."

But once people accept the idea that they are going to do something they never considered a possibility before, they need to know exactly what steps to follow. Otherwise no real action can take place. We're now ready to discuss the concrete actions that will enable a Contextual Design project to get started. We will deal with team formation in a later section; here, we will describe how to set the focus for a project, how to plan who to talk to, and variations on the data-gathering process that may be required by different problems.

SETTING PROJECT FOCUS

Before you can do useful work, you must define the problem you intend to solve in terms of the work you plan to support. Typically, a project's mission is defined in terms of the solution it will deliver: "an ordering system for all departments," "the next version of product X," "an electronic clipboard for doctor's offices." (As we discussed in Chapter 2, this is the kind of problem statement that is usually given to the project team by marketing or by the internal client.) To figure out what to do next—who to talk to and what to look for to decide what is important in this domain—the project team must transform this statement about the solution into a statement about the work.

Your initial project focus will usually be too narrow, too much restricted to exactly the work of the tool you expect to build. To see the whole work context and identify opportunities and potential problems, you want to expand the focus beyond tool use. Ask: What is the work we expect to support? How does this work fit into the customer's whole work life? What are the key work tasks? These are the aspects of work to find out about. Who is involved in making the work happen? Who are the informal helpers? Who provides the information needed to do the job, and who uses the results? These are the people to talk to. Where does the work happen physically? What is the cultural and social context in which the work happens? These constrain the interview situation you can set up. These questions will guide you in thinking about how your system fits into your customers' overall work. Use them to identify what kind of people you want to interview, what tasks you want to see performed, and what you want to watch for while you're there. Remember this is a focus, not a checklist. Use it to guide what you pay attention to during the interview.

Broaden your focus to include the whole work process

To expand your perspective on the work, look for metaphors for the work—unrelated kinds of work that have the same structure as the work you want to support. If you are studying online search and retrieval, you can study how people search for physical objects in libraries and grocery stores. This will help you understand the basic structure of finding, independent of technology and content. If you are studying PC maintenance groups, look at taxi dispatch services; the maintainers need to go out on calls without losing contact with a central organization in much the same way that a taxi is dispatched by the central office while maintaining contact with the office and with other taxis. Studying a taxi service would give insight into the problems of maintaining this kind of coordination and suggest different ways of organizing the PC maintenance group. Metaphors like this give you insight into the work you are supporting, suggesting hidden aspects that might be important. Use the metaphor to structure your thinking, and conduct

Study analogous work to stimulate insight into how work is structured interviews in the metaphor's work domain if it would be useful to know how it really works.

With a clear statement of project focus, you are ready to apply it to the particular project situation, starting by defining how to gather data. Different kinds of projects will constrain the data-gathering process in different ways: If you are extending an existing system, that system defines the work you need to study. If you are addressing a new work domain, you need to be open in what you study. The kind of data you look for will be driven by the work you plan to support, but also by the goals of the project.

DESIGNING THE INQUIRY FOR COMMERCIAL PRODUCTS

A project in commercial software may be generated in three principal ways. Each different starting point implies a different set of issues and a different way of collecting data.

Designing a known product: A "known product" is one of a class of products that is known and accepted in the marketplace, like a word processor or a spreadsheet. Competitive products are already established. The market has expectations for this kind of product—you must include certain capabilities to be taken seriously. This may be the next version of a product you are already shipping.

Gather data on people using competitive products. You must meet the market expectations they create. Gather data on the basic work practice of the market, whether the customers use competitors, your products, or no automated systems at all. Use your existing customer feedback channels to help set your focus. This will reveal what aspects of work are currently not well supported. Designing your product to support these unmet needs will differentiate your product from the rest of the market. If they are important enough, you will define the new field of competition for the next generation of products, just as the formatting capabilities of early versions of Lotus 1-2-3 defined the new ground of competition for spreadsheets. At the same time, gather data on detailed tool use. You want to make sure that you do the expected function just a little better than anyone else. You also want to pay

Look for the new delighters: the unrecognized needs attention to what aspects of existing products get in the user's way, and design ways to streamline it.

Addressing a new work domain: A new work domain is totally new. It has been created by changing work or life practice (the fitness industry) or new technological possibilities (telecommuting) and is not addressed well by any product. Any new product will change the way people work in the market, and there's no existing product to use as a guide. The danger lies in thinking that because the work will be changed, there's no way to study it. Before spreadsheets were invented, people did the work—they used paper ledgers to chart their accounts. Before word processors were invented, people did the work—they used typewriters. Define the work your new systems will replace, and study it to learn what matters and how it is structured so the market can make the transition to your new products. (This will not stifle any innovation in your products. Both the first spreadsheets and the first word processors were developed through detailed understanding of the people in their prospective markets.) Define the intent people are trying to achieve. Gather data on people achieving their intent with current tools. Look at how they use paper, informal contacts, and whatever else is available to do what they need to do. Look for problems and places where the lack of tools keeps them from trying to achieve their real intent. Use metaphors to think about what may be important in the new work domain.

All work is already being done some way; study it for clues

The new market may be best addressed not by a single product, but by multiple products working together to support the work comprehensively. When we discuss designing the system in Part 5, we'll show how to manage multiple coordinated products.

New technology: Sometimes a project seeks to take advantage of a technology that has just become available or affordable. Instead of being tied to a particular work domain, the project is looking for opportunities to use the technology. You may define specific products, you may design alterations to existing products to take advantage of the new technology, or you may discover that whole new markets open up once the technology is available.

Look for analogs of the technology and how they are used in the real world. If you are automating something that already exists, such as sound or text-to-speech, look for places in everyday life where sound or speech is already used effectively. Look at the context: What else happens when people talk, such as eye contact and nonverbal cues? When is silence important? Look at what the new technology replaces: for example, infrared links replace signal-carrying wires, so where are wires used? Network wires, control pad wires, speaker wires. Look for the underlying metaphor of the new technology and study that: a PDA (personal digital assistant) is like a Day-Timer with smarts, so look at Day-Timers and ask what you could do with them if they were smart. Look at the fundamental new characteristics introduced by the new technology: Wireless links allow moving around, so how is movement important? PDAs are small, so how does size matter? And use metaphors for the technology to get a different perspective of its use. Go to the places where the new technology can make a difference to stimulate your thinking about how it might be used.

Build on how analogs of the technology are used in the real world

Designing the Inquiry for IT Projects

IT projects tend to be driven by business needs. However, the statement of need tends to focus on the immediate problem as perceived by the customer. Responding only to the stated problem usually results in a patchwork of small systems, each addressing a small part of the work in isolation, and none working well with any of the others. It's often necessary to negotiate the project focus with the customer so that customer needs are met but the resulting system also ties work together. The proper role of IT is to work with the customer to step back, determine the underlying issues that resulted in this problem, and work out a solution that ties the work and the information systems that support it together. IT organizations always want to create and deliver coherent systems that work together to support a business seamlessly. Any new system should be defined to fit into the overall business strategy. Tying the work together means IT organizations always want to be in the business of process redesign. Rather than automating whatever idiosyncratic work practice exists, IT benefits from working with the customer to imagine changes to their process that take advantage

IT's role is to tie the work together through information systems of technology. There are three kinds of requests IT usually has to deal with.

Upgrades: The request is to add or modify a feature of an existing system. Typically this is called "maintenance" by the IT department. We avoid this term because "maintenance" implies that no new, interesting work happens in this task. In fact, much of IT's workload is in this kind of "maintenance," and much of the improvement or degradation of the information systems taken together is the result of "maintenance" work. So we borrow a term from the commercial vendors and call these "upgrades." The upgrade request is often stated in terms of a design change: "Just make it so I can enter several orders at once." Your challenge is to understand the reasons behind the request and design a solution that fits the need, keeps work practice coherent, and preserves the integrity of the system design. Look at the whole of the work task and related tasks to understand how the change affects the work as a whole. Look at detailed tool use to see what UI mechanisms work and which get in the way. Look for other point requests that can be addressed with the same mechanism.

Look at tool use and its edges to extend the system

New systems: The problem as stated is to provide a system to support some aspect of the business (e.g., order processing). There is no explicit intention on the client's side to change the way they work in any major way. Introducing a new system to automate the inefficient ways that things are done currently is a waste. The challenge is to move the design team and the client together to invent ways to improve the work. The result will be to define new ways of working and the software systems that support them. Expand your statement of focus by looking at the whole work process that the original request is a part of. How does it support the real work of the department? If this is the primary intent of the process, look at how the intent is accomplished. If not, ask what the intent is and whether it can be accomplished in a more direct way. Is the process contained in one department, or does it span departments? Plan interviews with people at each point in the process.

Ask: how will the new system support the real work of the department?

Process redesign: The project is started to implement a business process reengineering directive. Typically the directive does not specify exactly what the new work practice will be or the exact requirements on supporting systems. Instead, it just gives broad outlines of the new process and hints of supporting systems. "In the new claims-handling process, one person will be responsible for the claim from the time it comes in until it is settled. All claim data will be available to all parts of the company through a central database." The directive leaves open how the claims process works on a daily basis, how people will interact with the new system, and exactly what kinds of interactions the new system must support. The focus for such a project needs to look at the customers of the new process: what do they need, and why? Look at how the work is accomplished now: What have people had to do to make the process work? What will get in the way of introducing a new process? Helping people accept and adapt to the new way of working is a part of the design problem. Plan how to include the customers in the design process. When they are a part of redesigning their own lives, they will more easily accept and adapt to changes.

CD develops the details of business process redesign

The project focus gives the team an initial cut of what they are working on, who their customers are, and what the key tasks are. It suggests things to look for in the field and suggests some of the places to go. This prepares you to determine the specific interviewing situations needed to get the right data and make the project work.

Designing the Interviewing Situation

Your initial inquiry into the work gave you a focus for the project and also revealed some characteristics of the work domain and told you what work tasks you need to observe. Exactly how you will set up the interviews is driven by the nature of these tasks. The key questions for defining the interviewing situation are always: How do I get close to the work? How close can I get? How do I create a shared interpretation with the customer? Different kinds of tasks make different demands on the interview.

Normal: A normal task can be planned, is performed in a reasonably continuous session, and can be interrupted by the interviewer.

Writing a letter, delivering mail, installing software, and writing code are all normal tasks. The interviewer can plan to be present to observe a normal task and can interrupt at will to understand it. Normal tasks can be studied through a standard contextual interview. It may be useful to ask the customer to save work of the sort you want to study to do during the interview. This does alter the normal work flow, but very minimally, and the increase in relevant data makes it worth it. Audiotape these interviews, but videotape is rarely worth the extra trouble. Videotape them only if the work is so UI-intensive that you have to see the interaction to understand what's going on, or if it's especially important to communicate the customer experience to developers who can't go on interviews themselves.

Use a standard contextual interview

Intermittent: An intermittent task happens at rare intervals over the course of a day. It cannot be scheduled and does not last long. It's so infrequent that the chances of observing it during a standard contextual interview are low—you'd spend hours to get five minutes of data. Looking something up in documentation and recovering from a system crash are intermittent tasks. The key to learning about them is to create a trail that will enable the user to re-create a retrospective account of the event. In documentation, you could ask the user to keep a paper log of every time they use the documentation, perhaps numbering the pages themselves so they can walk through the story later. You could design the documentation so the user can keep their log right in the documentation itself. You might instrument online help, so the software automatically records what the user did. Start with a face-to-face interview, then leave them to log what they do. Return later to perform an interview that follows the form of a retrospective account, walking through each artifact in turn to discover what the user did.

Create a trail to walk and talk with the user

Uninterruptable: Some tasks simply cannot be interrupted to do the interpretation. A surgical operation, a high-level management meeting, and a sales call are all situations that cannot be stopped to talk about what is going on. In these situations you want to capture the events clearly enough that you can recall all the details later. You might plan interruptions, such as providing for regular 15-minute breaks in a long meeting where participants can

Plan discussion breaks between events

discuss what happened in the part of the meeting just concluded. You might videotape the event, then review the videotape with the customer, stopping to discuss events as they occur. If even videotape is too intrusive, you can at least keep good notes and review them with the customer. If you videotape, interpret the tape with the customer. You lose too much insight and cannot be sure of your interpretations if you review the tape alone later.

Extremely long: Some tasks take years to complete. Shipping a major software system, developing a new drug, and building a 747 are all tasks that take substantially longer than the two to three hours of a typical contextual interview. To understand tasks of this sort, pursue two strategies: first, interview a wide range of users at different points in the process and playing different roles in the process. Since work strategy repeats, common patterns will emerge even though the cases are different. Then, choose willing customers with the best examples and do a work walkthrough, which is like an in-depth retrospective account. Set up an event in which customers bring in project documentation from all parts of the process and walk through the history of the project, week by week, meeting by meeting. Use the project artifacts to ground the inquiry. Include project documents, such as plans, reports, and designs, and also process documents, such as the calendars and email of those most concerned. Use the artifacts to drive the conversation. Expect this recreation to take a day or two.

Create interviewing situations that reveal a cross section of work

Extremely focused: Sometimes the problem is so focused on the minutia of a person's actions that it's too hard to run a standard interview. You might be polishing the detailed interaction of a computer user with an application's UI or studying the details of how a craftsman manipulates his tools. You would miss too much if you depended on unaided observation, and you would also get in the way of the work too much if you interrupted every moment. This is a case where videotape can be useful. It will capture the details you would miss, and you can run it repeatedly until you understand a particular interaction. But view it and interpret what you see with the user. You cannot understand all their motivations on your own.

Videotape and interpret with the user

Internal: Sometimes the inquiry needs to focus on internal mental processes, such as how decisions are made. In this case, the interviewer must be present when the mental process is happening because there's no way to recover enough in a retrospective account. You may need to create events that will cause the mental process to happen so that you can be present. Then interrupt a lot; make a lot of hypotheses about what the customer is taking into account in their thinking. Warn the customer this will be very disruptive, but as long as the customer has to make the decision, they will keep working through it and you will learn something about how they do it.

Use ongoing observation with lots of interruption

Deciding Who to Interview

At this point you know what you are looking for and you know how to set up the interview for the tasks you need to observe. Now you must start putting names on the customers you will visit. In general, you want to interview two or three people in each role you identified as important to the focus. You want to collect data from 10 to 20 people in all, unless the focus is very narrow. Six to ten interviews is sufficient if there is only a single role or you are studying detailed UI interaction instead of overall work process. If you are making commercial software, you want to go to at least four to six businesses to see variety. In choosing sites and individuals, go for diversity in work practice. You are looking for the common underlying structure that cuts across your customer base. You will do this best by studying very different customers, rather than studying similar customers to confirm what you learned.

Diversity in work practice usually is not equivalent to diversity in market segment. Financial institutions, high tech, and retail may be different market segments, but office work is done very similarly in any modern corporation. These different types of companies will not give you substantially different perspectives. In fact, office work is so similar it is actually hard to get a different perspective. One design team studied the military and Japanese companies, in an attempt to find cultures that would be substantially different; they found little that was new. To get different work practice, look for different business strategies (doing the work as a business for hire vs. doing it as a department in a large company). Look for cultural differences (a trucking company vs. a high-tech

Interview customers whose work is as different as possible company). Look for different physical situations (a company distributed across several states vs. a company located at a single site). Look for differences of scale (a small business vs. a large corporation). If your customer is internal, see if you can study similar work practice in other companies. Look for other places in your own company where similar work is done, and study it. Use metaphors to give you different ways of thinking about the work.

Given these parameters for numbers and diversity, choose the people you will interview. It's okay to be smart when choosing—include the important client who has to buy into an internal project. Focus on customers from the key markets you think are most likely to spend money.

Let focus changes drive customer selection

Expect setting up customer visits to take a couple of weeks, by the time you've found the right person to interview, talked to all the people who are affected, and have set everyone's expectations correctly. However, don't get too far ahead in lining up the visits. As you study the data, you will change your idea of what to find out about next. You don't want to be locked into studying ten documentation writers after you've studied three and discovered that, for your purposes, they all work in much the same way. Make sure you talk to the people you will interview individually in advance and that they understand what will happen.

Your inquiry into the work that the project supports will yield lots of detail about the work and what to look for. It will be too much for anyone to keep track of during an interview. So boil it down to a short statement of the key characteristics of the work. This statement can be written by interviewers in their notebook and will keep them on track during an interview. A focus for an ordering system might be "how people find out about, decide on, and make requests for the things they need to do their work." Such a focus implies things to look for during an interview: "how people learn about what is available, through catalogs, friends, and local experts, whether formal or informal; who is involved in the decision and how they come to agreement; what processes have to be used to make the request and who gets involved in filling it."

A pithy focus statement keeps the interview on track

The initial focus will be revised and expanded through inquiry into the work. (In the above example, the team discovered that it matters to people to track the requests they have made and when they are expected to be filled.) Focus statements are best when they use simple language. People looking for "requests" will think more broadly about what a request might be and how it might be filled than people looking for a formal-sounding "order." The result will be greater insight into the work and consideration of a greater range of possible solutions.

MAKING IT WORK

For commercial software and internal systems alike, the crucial first step is to ground the design in relevant customer data. This part of the book has given you a solid grounding in the basics of setting up and running a successful interview. This way of collecting customer information is new, and most organizations do not have the procedures in place to make scheduling these interviews easy. The groups that have the easiest time are those who already create events with individual customers, such as usability tests or focus groups. There can be internal resistance, too. The sales force, marketing, or the internal customer representative can be suspicious of letting engineers talk directly to customers. (See Chapter 20 for strategies on dealing with resistance.) But reactions to the visits are nearly always enthusiastic. Customers feel like they are being listened to for the first time, and the sales force and marketing soon come to recognize the benefits. When the customers are internal, they feel like they have control over the new system. Teams developing custom software often do more interviews than strictly necessary to allow everyone to participate.

Customers feel heard and valued after an interview

As with all skills, experience comes with practice, but you need neither experience nor practice to get started. Whether you are working on the initial requirements for a large system or are refining the UI of a small system, you can define a data-gathering strategy appropriate to your project. A few interviews run along these lines will return a wealth of data on the customers you serve and the work they do. Increased interviewing skill will come with experience.

But be warned: it's addictive. People who get used to having contextual data when they design often have a very hard time breaking the habit.

Work Models 6

Each of the five types of work models has its own concepts and symbols representing one aspect of work for design. The five models were developed over time to meet the needs of the design problems we encountered. They represent the key aspects of work that design teams need to account for in their designs. We have found these five to be necessary to almost every problem and sufficient for most.

Work models are first built to describe work from the point of view of the one person interviewed. They do not and are not intended to represent everything that a person or his organization does. Each interviewer learned about some part of the customer's work as it related to the project focus. They also learned something about the work of the organization, as understood by this one customer. The first models we build represent this *individual* perspective. We even use conventions to show which parts of a model are built from the customer's actual experience and which represent the customer telling us how his organization is supposed to work.

THE FLOW MODEL

To get work done, people divide up responsibilities among roles and coordinate with each other while doing it:

> A rush order comes in. The woman who receives it calls the person responsible for filling it and mentions, in passing, that a rush order is on the way. The rush order will be shipped on time only because of her informal advance warning. When a new order-processing system is introduced, it does not allow this advance warning and rush orders start shipping late.
>
> A purchasing department is responsible for paying invoices as they come in. But they don't know if the goods were

actually received; they have to figure out who received the goods, send the invoice to him for approval, and pay it only when he returns it signed. Making the purchase and paying for the goods have been separated from the actual work of the organization. Formal sign-off and review processes keep the system working. The purchasing department gets so involved in maintaining these formal processes that they cannot handle finding vendors and making purchases well.

A specialist in another organization gets ready to produce a report. In times past he would have had a secretary type in and format the report; these days he not only creates the content, but he also defines the formatting and layout, checks spelling, and proofs the document as well. He has more control over the document in his own hands, but it's not clear that it's cost-effective for a highly paid professional to do basic stenographic and editorial tasks.

In each of these cases, the key issue is how people's roles are defined and how they communicate to get the job done. The *order receiver* had to communicate with the *order processor* to get rush orders accomplished on time; the *invoice payer* had to communicate with the *goods user* to find out if the invoice should be paid; the *content provider* became the *page designer,* instead of handing the content off to a secretary who could have played that role. All work in this world involves other people to some extent. Books are written for an audience, based on sources, submitted to reviewers, and passed to publishers. Code is written by developers for its users, from requirements, tested by a testing group, marketed by a product marketing group, and distributed to customers. Departments exist because a single person alone can't get the work done; the work must be broken into parts, which then must be coordinated. Different departments coordinate the different parts of the work, and people within a department coordinate to get its work done. The flow model represents this communication and coordination necessary to make work happen.

No real work happens in isolation

RECOGNIZING COMMUNICATION FLOW

Work flow (Figures 6.1 and 6.2) defines how work is broken up across people and how people coordinate to ensure the whole job gets done.

FLOW MODEL DISTINCTIONS

The *individuals* who do the work. In the consolidated models, the roles they play (see Part 3 for a discussion of consolidation). Each person or group is shown as a bubble. The interviewee's bubble is annotated with user number and job title. Everyone else's bubble just has job function.

The *responsibilities* of the individual or role. This is a list of what is expected of them—"coordinate schedules of all managers," "ensure samples are processed in the shortest possible time." Every bubble and place on the flow model is annotated with responsibilities.

Groups, sets of people who have common goals or take action together. Outside people may interact with the group as a unit, without knowing any individuals in the group. They say things like "I sent it to purchasing"; the particular person in purchasing doesn't matter. Groups are represented when a person has the same interaction with all its members. We may also show the interaction between a group member and the group as a whole.

The *flow,* the communication between people to get work done. Flow may consist of informal talk and coordination, or it may consist of passing artifacts. Flow is shown as arrows between individuals.

Artifacts, the "things" of the work, which are thought of and manipulated as if they were real. An artifact may be physical, such as a document or message. It may also be conceptual; for example, if a design conversation is thought about as though it has members, a history, attributes (public or private), and an existence separate from any one member or topic, it may warrant representation as an artifact. Where appropriate, the *mechanism* is shown—email vs. paper, for example. Artifacts are shown as small boxes on a flow.

The *communication topic* or *action* representing the detail of the talk or coordination represented by a flow. These are actions as opposed to artifacts, such as talk to set up meetings, arranging for review, asking for help. Examples might be "question about the system" or "request for help." Communication is written on a flow without a box.

Places that people go in and out of in order to get their work done, if it is central to the work of coordinating and collaborating. This is often a meeting room or communal space such as a coffee area. It is shown as a large box annotated with name of place and responsibilities.

Breakdowns or problems in communication or coordination, represented as a red lightning bolt (black in this book). ❑

How do job responsibilities get assigned to people? What are the different roles people take on to get work done? How do new tasks get passed to a person? Who do they get help from? Who do they have to work with to accomplish their tasks? How do they use physical places and artifacts to help them coordinate? Who do they give the results to and in what form? Work flow is the rich pattern of work as it shuttles between people, the interweaving of jobs and job responsibilities that gets the work done. Work flow represents every phone call between

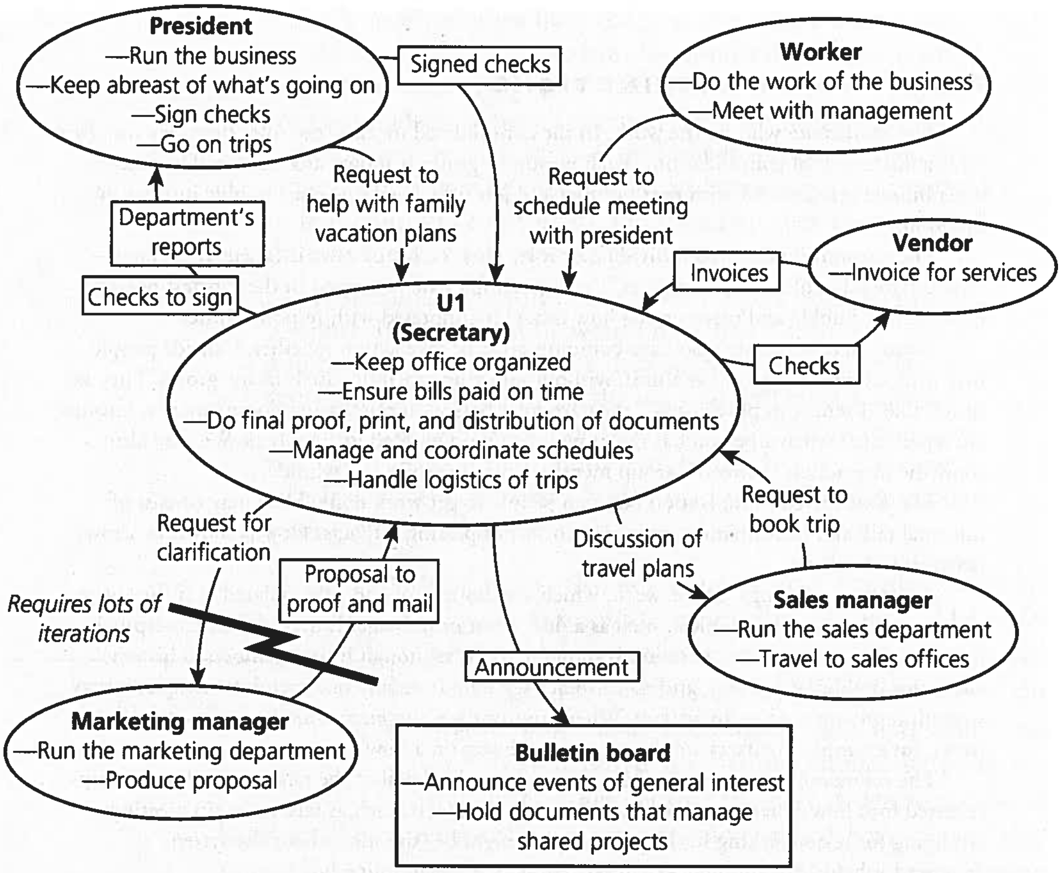

President
—Run the business
—Keep abreast of what's going on
—Sign checks
—Go on trips
Signed checks
Worker
—Do the work of the business
—Meet with management
Request to help with family vacation plans
Request to schedule meeting with president
Department's reports
Vendor
—Invoice for services
Invoices
Checks to sign
U1 (Secretary)
—Keep office organized
—Ensure bills paid on time
—Do final proof, print, and distribution of documents
—Manage and coordinate schedules
—Handle logistics of trips
Checks
Request to book trip
Request for clarification
Discussion of travel plans
Proposal to proof and mail
Requires lots of iterations
Sales manager
—Run the sales department
—Travel to sales offices
Announcement
Marketing manager
—Run the marketing department
—Produce proposal
Bulletin board
—Announce events of general interest
—Hold documents that manage shared projects

FIGURE 6.1 Secretarial work. This flow model is typical of secretarial work. Secretaries often act as the center, the hub, of a department. In this model, we see this graphically in the many lines that diverge from the central bubble. We can see the great diversity of the hub function in the many types of communication on the lines—everything from formal reports being passed up the management hierarchy to informal requests to smooth the personal lives of people in the department. The accretion of hub responsibilities in one person is natural; once a person is coordinating one aspect of an office, it is natural for them to coordinate other aspects as well. From this diagram we see the nature of hub work—lots of different activities, communication with lots of different people, lots of interruptions, and lots of tasks going on at the same time.

two people, every document passed for review, every email message, every conversation between people in the hall. These are all instances of passing an artifact, communicating information, or coordinating to

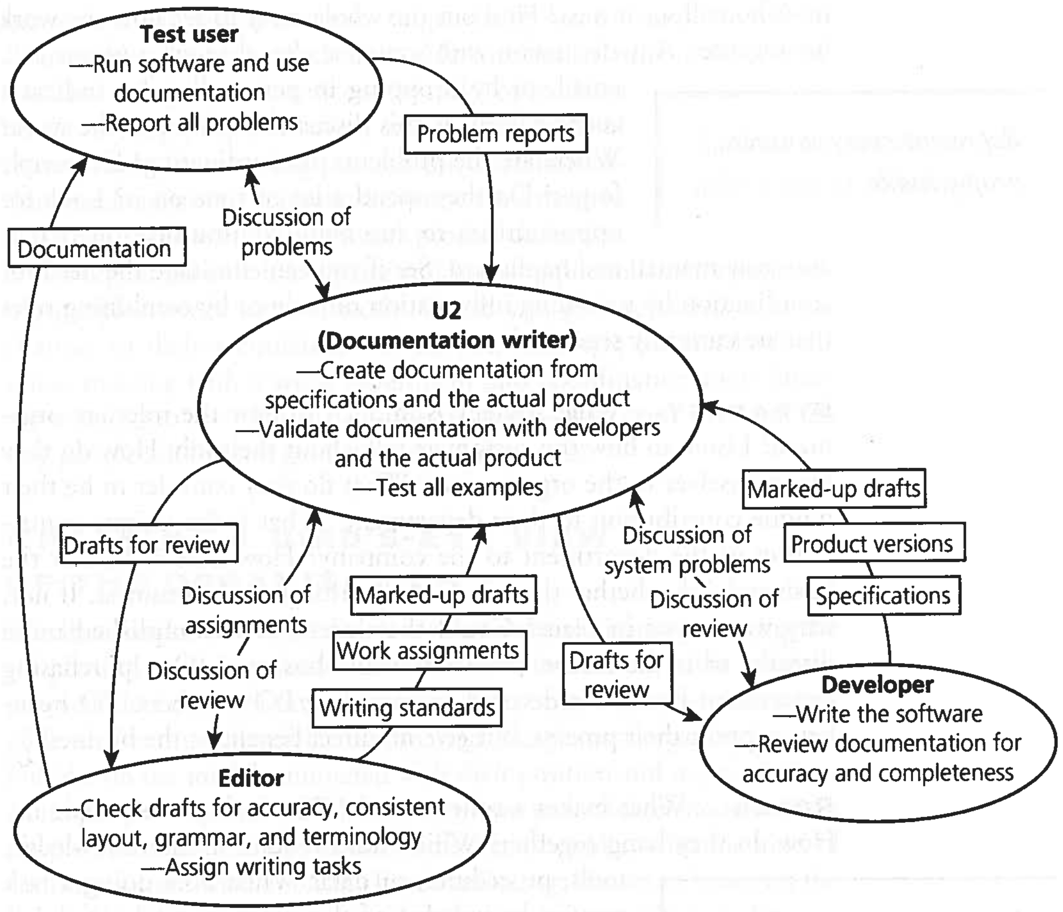


FIGURE 6.2 Creative work. This flow model is typical of creative work. We see communication with those who depend on the work and with those who assist in the creation. But most of the interactions are focused on the task of creation. Compared with the "hub" type of job, this work is much more continuous and coherent.

do a job, whether as part of a formal process or as an informal way to get the job done.

When people coordinate through email or paper, it's easy to see. It's harder to see how casual conversation and handwritten notes support work flow. Here's what to watch for in an interview.

COORDINATION. Any artifact received or handed on indicates coordination with someone else. Where did it come from? Who created it? Who will see it next? Find out the whole story to see how the work fits together. Any discussion with someone else, through a phone call, email, or by dropping in personally, also indicates coordination. Is this discussion critical to the work? Where are the problems in coordinating? Do people forget? Do they spend a lot of time on it? Look for opportunities to automate communication that is currently manual and haphazard. See if you can eliminate the need for coordination by providing information directly or by combining roles that are currently separated.

Represent every contact people make

STRATEGY. What strategy is implicit in how the roles are organized? Listen to how the customers talk about their job. How do they see themselves in the organization? What do they consider to be their unique contribution to their department? What is the unique contribution of the department to the company? How does it further the business? Ask whether the role is really critical to the business. If not, why was it put in place? Could that intent be accomplished more directly, or is the intent irrelevant to the business? (One purchasing department has a role devoted to providing PO numbers. PO numbers support their process, but give no direct benefit to the business.)

ROLES. What makes a coherent role? Watch the tasks people do. How do they hang together? Which tasks require similar knowledge, tools, procedures, or data? When does doing a task require knowledge of the progress made in doing another task? These tasks tend to be performed by the same role. Technicians, for example, need to know the history of a problem and of prior attempts to fix it in order to serve the customer well. If problem calls are handed out to the first available person, regardless of history, service will be poor.

Note what responsibilities people take on—even responsibilities that are not part of their jobs

INFORMAL STRUCTURES. Look at the ways people go beyond the formal structure: A secretary becomes known as the expert on creating forms. Soon whenever anyone has a particularly difficult form to create, they pass it to her and she does it for them. A scientist has special instructions to communicate to her lab technician. She writes a note on a materials tracking tag, knowing he will see it. A manager has to assign resources to get things out on time. He invents a status meeting to get it all done. He consciously runs it like a combination bingo game and commando operations center to keep people involved and excited. Each of these people is inventing process and communication mechanisms to support the work they need to do. They show where the formal process definition of the organization is inadequate and reveal opportunities for supporting people's needs more directly. Could you give scientists a better channel to their technicians? Could you eliminate the need for the status meeting with a work assignment and coordination tool? Study the meeting to see what the tool needs to do—and don't overlook the way people ask for and get help around the edges.

Look at the actions people take without thinking

Creating a bird's-eye view of the organization

The flow model offers a bird's-eye view of the organization, showing the people and their responsibilities, the communication paths between people independent of time, and the things communicated—either tangible artifacts or intangible coordination. People and organizations are bubbles on the model, annotated with their position and responsibilities (roles are not represented directly until we consolidate models across people). Flow is indicated as arrows between bubbles, with the kind of communication written on the line. Artifacts are shown in boxes on the line; informal communication and actions are written without a box.

Where places such as meeting rooms or virtual places such as shared areas support communication, the flow model shows them as well. When a place is important to coordination—meeting rooms, bulletin boards, and shared drop-off areas—they appear as large boxes at the end of a flow. Just as individuals are annotated with their responsibilities, places list their responsibilities in supporting communication and coordination. Automated systems and databases usually should not go on the flow. The only exception is when they are acting like a physical place or like an automated person, and they are critical to coordination between people. Then they are shown as a large box with responsibilities.

Represent locations, things, and systems when they make a place to coordinate

When communication breaks down—people don't get something they should have received or don't respond when a response is needed—we show the problem with a lightning bolt.

Do not limit the model to the formal definition of how work is supposed to be done. The defined process of the organization is not a good guide to how work is actually accomplished. Every day, the people in the organization design how their jobs will really be done. As they encounter problems and obstacles, they create solutions, and the solutions become part of the real work. The flow model needs to capture how work is really done, including all the informal interactions that make it work. From this representation, you can find good work practice to incorporate into a system, identify problems to eliminate, and see the pattern of communication a system must allow for.

The real interactions between people reveal glitches in the work

THE SEQUENCE MODEL

Work tasks are ordered; they unfold over time. But the steps people take aren't random; they happen the way they do for a purpose:

> A man reads a mail message and, after replying, saves it in a folder called "Phone book." He'll never need that message again. He's just saving it because it has the sender's telephone number on it, and it's a convenient way to look it up. So telephone numbers matter even when email is the primary form of communication, and telephone calls may be triggered by email. Anyone trying to build the complete personal organizer can build on this to tie phone contacts and email together.
>
> A woman paying her bills first gets out her checkbook, bills, paper record of accounts, envelopes, and stamps; then records the amount of every bill and makes sure she can pay them all; then writes each check in turn; and then puts each in an envelope and addresses it. So the stages of paying bills are collect and organize; plan what to pay and how, making sure not to overdraw the account; actually pay the bills; and put them in envelopes to send out. A home accounting program can build these steps in directly.

> A scientist is interpreting the results of an experiment. He puts the raw numbers in one column, then in each successive column shows the result of one transformation. He needs to see not just the final result, but the process by which those results are achieved. An analysis tool that hid the calculations, and only revealed the result, would not be acceptable.

The actions people take in doing their work reveal their strategy, their intent, and what matters to them. A system that builds on these can improve the work they do. Understanding the real intent is key to improving work practice; you can redesign, modify, and remove steps as long as the user can still achieve their underlying intent. An intent is stable—for example, people have had the intent of communicating over a distance for ages. The steps, the way that intent has been achieved, have changed over time—from handwritten messages to the telegraph, the telephone, and videoconferencing. Supporting the current work steps just automates the way things are done currently (and because paper is almost always faster than computers, if the system does nothing but automate existing steps, it almost always loses). The goal is to change the work steps to make work more efficient. But the system must support all the intents concealed in the work, not just the primary espoused intents. If users have an intent of planning how to pay bills before they start writing checks, and the system doesn't support planning, the system will not be accepted.

Understanding customers' intent is the key to design

All work, when it unfolds in time, becomes a sequence of actions—steps to achieve an intent. A sequence model (Figure 6.3) represents the steps by which work is done, the triggers that kick off a set of steps, and the intents that are being accomplished. They are your map to the work that your new system will change. Sequence models supply the low-level, step-by-step information on how work is actually done that designers need to make detailed design decisions. The sequence model is most similar to flow diagrams or task analysis (Carter 1991), but is unique in stating the intent and trigger for the sequence. A sequence model starts with the overall intent of the sequence and the trigger that initiates it. Then it lists each step in order, at whatever level of detail the interviewer collected. Any steps that cause problems are labeled with a lightning bolt. When modeling the work of an individual, the

From any one person's point of view, all work is a series of actions

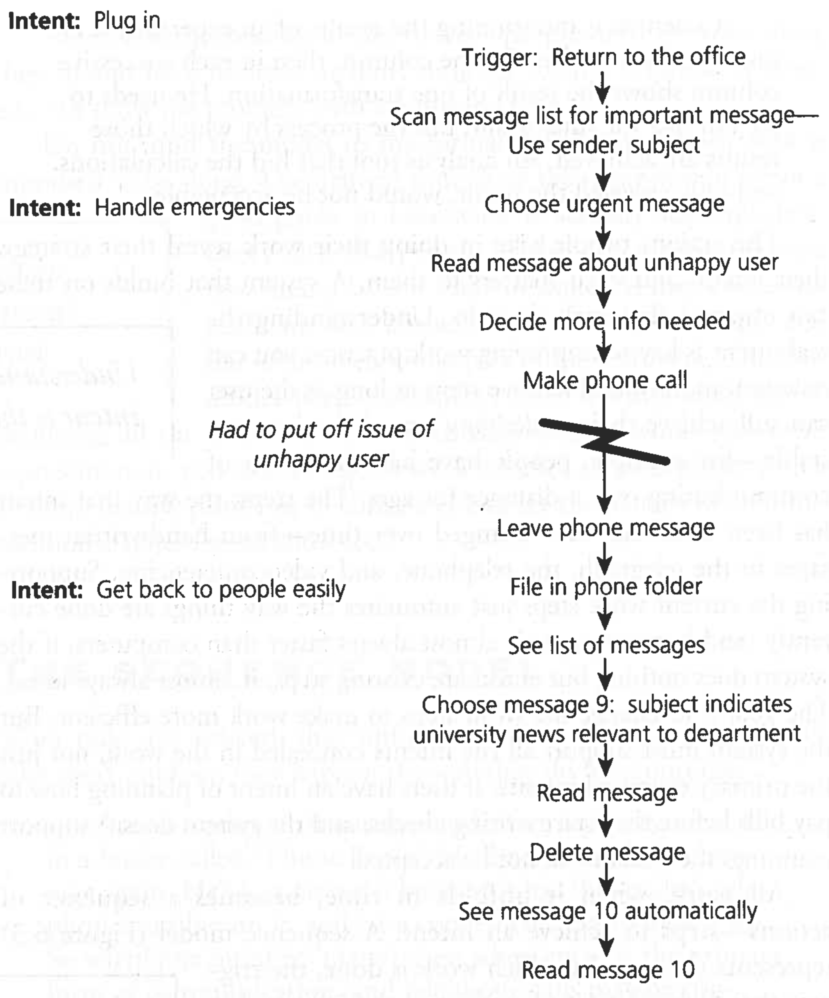

Intent: Plug in
Trigger: Return to the office
Scan message list for important message— Use sender, subject
Intent: Handle emergencies
Choose urgent message
Read message about unhappy user
Decide more info needed
Make phone call
Had to put off issue of unhappy user
Leave phone message
Intent: Get back to people easily
File in phone folder
See list of messages
Choose message 9: subject indicates university news relevant to department
Read message
Delete message
See message 10 automatically
Read message 10

FIGURE 6.3 Sequence model for handling mail. This sequence model shows how one user handled mail on one specific day. The intent is stated at the top left: "Plug in." This conveys the nature of handling mail for this user: much of his communication is through email, and when he left his office, he separated himself from this communication. Returning and checking mail was a reconnection, a "plugging in." This is implied by the trigger for starting this sequence, which indicates he does it whenever he returns to the office. The arrows indicate the sequence of steps. When he completed handling an emergency, he saved the message in a folder he uses as a phone book. This action indicates an unrelated intent, keeping a contact list up-to-date, which he handles opportunistically.

sequence model does not attempt to show pattern or repetition; we identify those when we consolidate. Sequences may be studied at any level of detail, from the high-level work to accomplish an overall task to the detailed interaction steps with a particular user interface.

Collecting Sequences During an Interview

Collect sequences in an interview by watching people work or by getting a detailed retrospective account of their work. The hardest thing about seeing sequences is knowing what to pay attention to, and this changes depending on the project focus.

Steps. If you are studying the work across the department, or if you are learning about a new market, you'll collect sequences at a fairly high level of detail. You want the actions people take, but not necessarily broken down into each movement. So writing a letter might look like: Get project information from project manager. Extract deliverables and delivery dates important to the customer. Write

Sequence Model Distinctions

The *intent* that the sequence is intended to achieve. Secondary intents will be embedded in this primary intent, and they are named as they are identified.

A *trigger* causing the sequence of actions. It is the notification to the user to take action. Triggers we have seen include the height of a stack of paper on a desk, the arrival of mail, receiving a request, and seeing a misplaced line of text in a document.

Steps, the action or thought preceding an action. In an actual sequence model, a step represents what actually happened. As we step back from the actual steps and look for purpose and strategy, the steps become more abstract. They move away from specific behaviors toward fundamental purpose.

Order, loops, and branches indicated by arrows connecting the steps. These reveal strategic and repetitive patterns of work. When the customer must make a decision about how to proceed, we show that as a branching path. The order gives us an access road map to ensure smooth transitions between tasks and allows us to see what steps could be combined or skipped without serious violation to the users' conception of what is going on in their work.

Breakdowns or problems in doing the steps shown with a red lightning bolt (black in this book). ❑

introductory paragraph describing current project state. Enter dates. . . . This level of detail shows the overall structure of the work and how it fits together without giving huge amounts of detail about each task.

If you are designing a system or tool, study the tasks the tool supports in more detail. Look at what people do and also *how* they do it. So writing a letter might look like: Scroll window to find last letter written. Open it. Delete all content. Save under new name. Enter name of recipient. Pull Rolodex closer. . . . At this level of detail, we see the structure of the task and the actions that make it happen.

Capture actions at the level that matters for your project

If you are designing the user interface, look at eye movement, hand movement, hesitations, everything. So writing a letter might look like: Use vertical scroll bar until icon for last letter written comes in view. Double-click on item to open. Read recipient name and scan first paragraph to make sure this is the right letter. Choose "Select All" from Edit menu. . . . This level of detail shows how the user interacts with the UI and reveals the issues for the UI to address.

In practice, the levels of detail blur somewhat, and it's safer to get more detail rather than less. Each action has a purpose in the user's mind. If it looks random to you, that's only because you don't know what the purpose might be. In a word processor, we repeatedly saw the user, with the cursor at the end of the line, hit the right arrow, see it move to the next line, then hit the left arrow to move it back. Even this was not random; he was checking to see if he was really at the end of the line or if there was extra white space because, in that word processor, the white space would make the line wrap.

Customers' actions are never purposeless

Hesitations and errors. Notice when the customer hesitates or makes errors. These are your clues to his thoughts. Intervene and ask questions to find out what he is thinking about. Hesitations and errors indicate places where the customers' understanding of work is being contradicted by the tools they are using. This is an opportunity for your system to do better. If a task is largely a thinking task, hesitations reveal decision points in the process.

Any glitch reveals a thought step

Stop the customer and ask him to explain what he is trying to decide at that moment. Try to get him to think aloud, to reveal more of the issues.

TRIGGERS. Every sequence has a trigger—the event that initiated it. Triggers may be discrete events, such as the ringing of a telephone, the arrival of an invoice, or a person arriving at the door. Triggers may be based on time, like the first of the month or the first thing in the morning. Triggers may be less tangible, such as the pile in the in-box getting too large. Whatever the trigger, if the work is automated, it must have an analog in the new system. The system needs a way to tell the user there's something to be done. Otherwise, the user won't take action—for example, one mail product simply gets slower the larger the in-box gets. This doesn't act as a trigger for the user to clean it out; it just makes the product more and more frustrating to use.

Watch how automation removes effective prompts to action

INTENTS. The intent defines why the work represented by a sequence matters to the user at all. Every sequence has a primary intent, which applies to the whole sequence. Then there will be secondary intents, which drive the particular way the work is carried out. So our bill payer has a secondary intent of not overdrawing her account and of redefining who to pay and how much to pay so that important bills are paid and the account is not overdrawn. Intents are usually identified after the sequence is written, when there is time to look it over and think about what lies behind the customer's actions.

Find the intents implied by the actions

Sequences capture the most basic information about work practice. Not only do they tell you how work is *really* done, they show how it is structured and the intents people care about. They present the detailed structure of work that designers will need when it comes time to structure the system. And they cut across the other models, tying them together. Because sequences are time-ordered, they show how different roles interact in different places, using artifacts to support communication and actions to get the work done.

Sequence models reveal the detailed structure of work

THE ARTIFACT MODEL

People create, use, and modify things in the course of doing work. The things they use become *artifacts,* like archaeological findings. They each have their own story to tell about the work:

> In one organization, a first-level supervisor prints the spreadsheet he uses to track projects weekly and gives it to his manager. His manager makes check marks against each project to indicate his approval and may make additional notes on the side. Then he signs at the bottom and gives it back. In this way the supervisor's personal tracking sheet becomes a sign-off mechanism and a way for the manager to communicate problems and issues. It suggests that sign-off and feedback are part of the job; an automated project-tracking system could build these features in.
>
> Another woman builds a spreadsheet to calculate end-of-year results. The calculations take 15 minutes to do—then she spends the next 45 minutes making the spreadsheet look good so she can hand it out at the next management review. When a spreadsheet is given careful formatting, it's clear that the way information is presented is an important consideration and that spreadsheets are presentation tools as well as calculation tools. The original spreadsheet tools only displayed text; they were replaced with tools that could do fancy fonts and gave full control over the look.
>
> Another organization has the goal of raising the level of cost consciousness among its people. They have a standard form for making a request for a purchase. The form has a place to describe the item and a place to justify why it's needed but no place to show the cost. When a purchasing form has no place to show cost, it suggests that cost is not a big concern in the organization. An automated purchase order request system could raise cost consciousness just by making cost prominent on the screen.

Artifacts capture traces of people's work practice

Artifacts are the tangible things people create or use to help them get their work done. When people use artifacts, they build their way of working right into them. The artifacts show what people think about when they work and how they think about it. An artifact reveals the assumptions, concepts, strategy, and structure that guide the people who work with it. Artifacts might be to-do lists, forms, documents, spreadsheets, or physical objects under construction (circuit boards, cars, airplanes). Artifacts may be bought, designed intentionally, or created on the fly. They are manipulated in the sequence models and passed between people in the flow model.

In their structure—how they are arranged into parts and the relationship between the parts—artifacts show the conceptual distinctions of the work. When displays showing the status of a network are separated from displays of trouble alerts, this indicates that tracking ongoing status is different work from responding to alerts. When notes are written on a presentation handout, not where there is white space to write them on, but jammed in next to the text they refer to, this indicates that the close spatial relationship of text and note matters to the writer. When the list of things that a person would like to get is separated from the shopping list, this indicates that a clear distinction exists in the person's mind between the nice-to-have-someday items and the I-will-buy-this-today items. An automated shopper's planner had better provide a way to track long-term possible purchases separately from today's shopping list (Johnson et al. 1988).

Artifacts make customers' conceptual distinctions concrete

An artifact model (Figure 6.4) is a drawing or photocopy of the artifact, complete with any handwritten notes. The model extends the information on the artifact to show structure, strategy, and intent. Highlight structure with lines and labels marking the different parts. Annotate the location of the parts showing how they are placed to give them prominence or support the artifact's usage. And write intents directly on the part of the artifact that supports the intent. Lightning bolts show where the artifact interferes with the work, whether because the defined structure does not match the work, because needed information is missing, or because it is too cumbersome to use.

Collecting Artifacts During an Interview

Artifact models always require interpretation to reveal their intent and usage. You can do this best with the customer during the interview. Look for and inquire into:

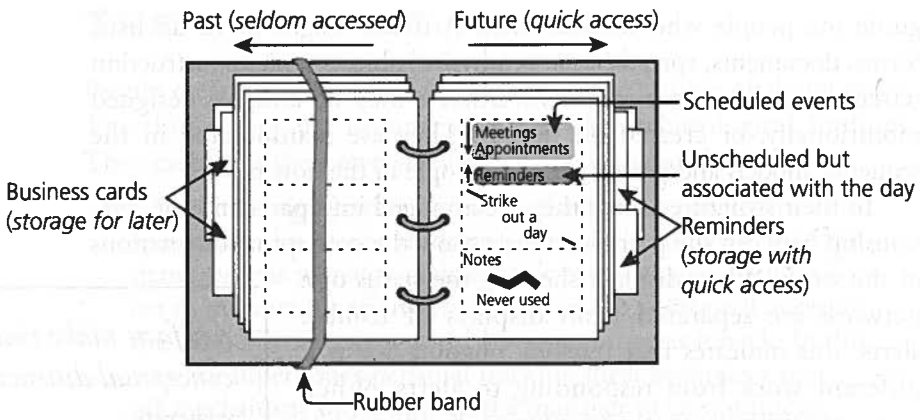


FIGURE 6.4 Artifact model. This physical model shows the structure of an artifact, in this case a personal calendar. The usage of this calendar reveals that it is not only about managing time; it is organizing an entire life. The rubber band makes the distinction between past and future. The calendar is acting as a storage place for reminders and to-do lists as well as a calendar. When the calendar gets too fat, this is a convenient trigger for dealing with the to-do lists. The usage of the day view shows additional distinctions: meetings are listed from the top of the day down, but reminders of a more general nature are written from the bottom going up. Reminders are attached to a day; they are not kept in the provided "notes" area, so it is not used.

STRUCTURE. All artifacts have structure, even the most informal. People naturally create a structure to represent their thought, even when they start from a blank page. If they didn't create the artifact on the fly, they may start from a given structure, either because it came as part of an artifact they bought or because they designed it themselves before starting the actual work. In this case, the structure inherent in the work wars with the given structure, and the artifact will show every place there is a mismatch. So the notes space on a daily calendar may be used for notes, but it may be left blank or used as a rolling to-do list. If everyone uses it like a to-do list, then organizing the day and scheduling are intimately intertwined.

Structure reveals how the work is organized

Look to see how the artifact is structured. How does the presentation—layout, fonts, formatting, and white space—reveal structure? Assume every grouping of information corresponds to a conceptual distinction in the customer's work. Can you and the user figure out what it is? Can you make these distinctions real in your system?

ARTIFACT MODEL DISTINCTIONS

Information presented by the object, such as the content of a form (e.g., a doctor's name, nurse's name, patient's name, and diagnosis).

Parts of the object, which are distinct in usage, such as page, kind of page (table of content vs. title page), headline, or figure in a diagram.

Structure of the parts explicitly in the object as given and implicitly in its usage: the division of a form into a section for the doctor's use and a section for the nurse's, the grouping of cells in a spreadsheet to represent part of the data for a single purpose, or the way some people use the top of a day within a calendar for meetings and the bottom for reminders.

Annotations, which indicate the informal usage of the object beyond that allowed for by its explicit structure: Post-its stuck to a document, highlighting, and notes written on the side of a report.

Presentation of the object: color, shape, layout, font, white space, emphasis, and how they support usage.

Additional *conceptual distinctions* that are reflected in an artifact and that matter in its creation and use: past, current, and future in using a calendar; structure and content that repeats in a report from month to month; x-height and caps height in page layout.

Usage of the artifact—when created, how used, how people move through the parts of the artifact.

Breakdowns or problems in using the artifact, represented as a red lightning bolt (black in this book). ❑

INFORMATION CONTENT. The content of an artifact is the information, specific to the work, that the artifact carries. The content of an artifact tells the story of a part of the work—how the content was put in, how it was used, and who used it. The content fits into the structure of the artifact—or it doesn't, in which case customers modify the defined structure. Seeing how the content is manipulated reveals the artifact's usage—how it supports the work and also the detailed interaction with the artifact in the course of working. So each meeting on a personal calendar suggests the story of the work task that the meeting supports, but it also suggests the detailed story of how the user interacted with the calendar to put the meeting on it.

Content is the trail left by real events

Look for the information the artifact carries and how it is used. Use the artifact to drive a retrospective account, as we discussed when describing interview principles in Chapter 3. Why is this artifact an appropriate carrier for this information? Who will see it and when? What would happen if the artifact didn't exist? Can you make the needed information available more simply in your system?

INFORMAL ANNOTATIONS. Informal notes and annotations are a gold mine of information. They tell you about the actual usage of the artifact. Did the defined structure get used? Was it extended? Was the artifact used to carry additional information by writing notes on it? Why was it used? What made the artifact the convenient carrier for the message? Can you put other channels in place to make this unnecessary? Can you see how the artifact didn't match the work, and can you see how to make your system fit the work better?

Annotations reveal usage and communication

PRESENTATION. Content and structure are revealed in the artifact's presentation. Look at formatting, the layout of parts on the page, and the use of white space. How does the artifact attract attention to some parts of the content and downplay others? The presentation supports the intent of the work if well designed and gets in the way if not. If the artifact is redesigned or put online, how should your system present it for easy interpretation in the same kind of way?

Presentation directs the eye and reveals importance

INQUIRING INTO AN ARTIFACT

There are two levels of inquiry into artifacts. The first is to see how an artifact supports the customer's intent. The presentation, content, and structure are all clues to what matters in the work. So notes scribbled on a materials-tracking card telling the technician how to handle the material show that direct communication between user and handler is important. Any system that interrupted the communication (such as an automated tracking system) would cause problems in the work. To be successful, such a system would have to provide another way to accomplish the same intent. At this level of inquiry, we look at structure and usage to derive intent, to show why the artifact matters and what any automated system needs to account for. (See Muller et al. [1995] for an example of such an inquiry.)

Walk through artifacts with the customer to see what they mean

If you think that the artifact might be supported or automated, then a detailed inquiry into the interaction with the artifact provides clues in how to structure the system. Things that cluster in the artifact are conceptual groups that should be kept together. The natural pattern of interaction with the artifact is a good guide to appropriate interaction with the system. So the notes on the materials-tracking card indicate that, if we want to automate materials tracking, we have to support informal communication between user and handler. This communication may happen at any time after the materials are received, so a single note that can only be entered when the materials are received won't do. Since the handwritten note is its own record, and having the record matters, the automated system needs to keep instructions related to the material available over time.

Bring back copies of used artifacts

Artifacts are the concrete trail left by doing work. They capture multiple stories of how work happened, making it possible to walk through a retrospective account of those events. As a physical object, an artifact makes the way customers think about their work tangible, so you can see and inquire into it. But artifacts do not speak on their own; collect examples that have been used and interpret them with the customer during the interview to reveal their meaning.

THE CULTURAL MODEL

Work takes place in a culture, which defines expectations, desires, policies, values, and the whole approach people take to their work:

> A vendor creates a product that helps development teams control their development process. The product is well designed and well made, but fails in its target market of UNIX shops. UNIX shops pride themselves in getting code out without needing a formal process.
>
> Another vendor makes an instrument so straightforward that unskilled operators can run it with ease. Their customer base won't buy it because they consider themselves highly skilled professionals who can run complicated systems.
>
> Another company gives their scientists software that simplifies the reporting of experimental lab results. The scientists

reject the system because they consider proper reporting of results to be part of the job of a scientist and don't want it simplified.

In each of these cases, there was nothing wrong with the system delivered. It was designed and built well and solved a real problem. There was no technical roadblock to its use at all. In each case, what prevented the system's success was the culture of its proposed users. If a system conflicts with its customers' self-image, or doesn't account for the constraints they are under, or undercuts the values important to them, it will not succeed.

Successful systems fit with their customers' culture

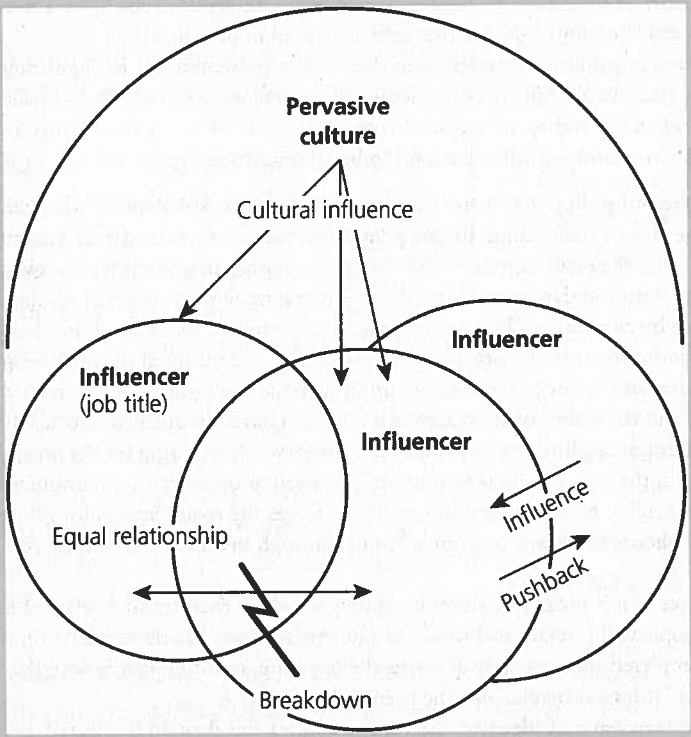
The *cultural context* is to us like water to a fish—pervasive and inescapable, yet invisible and intangible. Cultural context is the mindset that people operate within and that plays a part in everything they do. Issues of cultural context are hard to see because they are not concrete and they are not technical. They are generally not represented in an artifact, written on a wall, or observable in a single action. Instead they are revealed in the language people use to talk about their own job or their relationships with other groups. They are implied by recurring patterns of behavior, nonverbal communications, and attitudes. They are suggested by how people decorate and the posters they put on their walls.

Culture is as invisible as water to a fish

The cultural context includes the formal and informal policy of an organization, the business climate created by competitors and by the nature of the business, government requirements, the decor of the site, the self-image of the people doing the work, and the feelings and fears created by the people or groups in the organization. Culture influences work by altering the choices people make. Because they don't want to have to deal with a certain group, or because they consider themselves professionals, or because they are worried about what their competitors are doing, people change the way they do their work. Design teams that understand these constraints can build their systems to account for them.

The cultural model makes influences concrete

Cultural Model Distinctions



Influencers who affect or constrain work, shown as bubbles. These may be individuals or formal groups in the organization. They may be a collection of people who are not a formal group but are thought of together ("management"). They may be external influencers such as customers (and possibly multiple customer organizations), government regulatory bodies, standards groups, or competitors. They may represent the overall culture created by the organization or shared by the people doing the work.

The *extent* of the effect on the work shown by the amount the bubbles overlap. It suggests whether essentially everything about the work is affected by this influence or whether the influence is more partial. So the Food and Drug Administration influences the work of food and drug companies through its reporting and testing requirements, but this influence does not constrain everything about developing the food or drug product. On the other hand, everything an assembly line worker does is affected by the requirements of the assembly process.

Influence on the work. Arrows represent the direction of influence (who is primarily affecting whom) and how pervasive it is (whether this is an influence of one individual or group on another or whether it is more pervasive across an organization). We also represent pushback; in real situations it is rare that influence is all in one direction.

Breakdowns or problems interfering in the work, represented as a red lightning bolt (black in this book). Because all influences restrict work in some way, we only show breakdowns on the cultural model when they are especially harmful.

The following kinds of influence tend to be relevant to design:

Standards and **policy** that define and constrain how work is done or what can be used or bought, or the lack of such standards as a policy. So many companies define a standard PC configuration that they will support: "Use this configuration or you're on your own." Other companies live with standard procedures defined by themselves or imposed on them by the government or by customers: "Prove your process is compliant or we'll use another vendor."

Power, both formal in the organizational structure and informal through people's networks, expertise, and history. Power shows up in who has the right to decide who will do what in their work and the extent of autonomy a person can have. So one boss sets up his secretary's computer environment, limiting her ability to recover when anything breaks down: "I'll fix your machine in the way *I* think is important." In another organization, reimbursement for expenses is controlled by administration, which enforces the requirements for filling out paperwork and can choose to allow exceptions: "Jump through my hoops and I'll let you have your money."

The **values** of a company or team: what they stand for that produces a set of expectations about how people will interact and work. So one organization has the expectation that a project will be completed the same way as it was the last time, resulting in a feeling that innovation is unwelcome: "If it's a different plan, be prepared to justify it."

A group's own sense of **identity,** the way in which what they do is affected by how they think of themselves. So one UNIX shop held that they did not need to do formal up-front analysis and design because "we don't do process."

People's **emotions** about what they do, including fear about being laid off or getting in trouble for raising issues, or people's pride in what they do. So knowing that "email can be read by anyone, including management" led people in one organization to discontinue its use.

The idiosyncratic **style, values,** and **preferences** of an individual or team, creating a work environment that circumscribes others. So one boss will not use the computer, forcing his secretary to handle all his email communication: "Use the computer for me because I won't." Or a team can't work past 4:30 because everyone has outside activities that pull them away: "We are committed to home activities; schedule around them." ❑

RECOGNIZING THE INFLUENCE OF CULTURE

Culture is invisible, but can be deduced from things you see and hear.

TONE. When you walk in the door, what is the tone of the place? Industrial and sterile? Carefully designed and trendy? Formal and elegant? Messy and haphazard? When the customers design their workplace for elegance, they are unlikely to accept a system that looks haphazard. When they spend little time designing their workplace, just the bare minimum so that they can work, they are unlikely to accept a system that is overdesigned, which looks like time and money was wasted on elegance.

A valuable system helps people be who they want to be

POLICIES. What are the policies people follow, and how are they recorded? Are there policy manuals, and are they used? Do people wanting guidance on doing their work routinely check them? Or is the operational policy—the policy that affects work on a day-to-day basis—really passed by word of mouth? If so, how much is based on real directives, and how much is folklore? Is policy generated by fear of a regulatory agency, of another organization, or of a manager? You can hear policy in the words people use: "We won't buy anything but UL-rated power supplies. They had a non-UL supply catch fire over in building 10 a while back." If UL rating matters, you can highlight UL-rated equipment in the catalog you develop. "Better get these procedures documented properly. One of our competitors was cited for out-of-date documentation, and their stock dropped three points." If written records are an important part of the work, you can implement systems that maintain them. The policies that people care about point to problems you can solve.

A valuable system makes conforming to policy easy

ORGANIZATIONAL INFLUENCE. Are there organizations, individuals, or job functions that keep showing up, either as troublesome or helpful? What are the organizations or job functions that always seem to get in the way? Who are the people who constantly show up as the ones who can solve the problem? Listen to how people talk about others: "Don't call maintenance about this. They'll take it away to check it out and you won't see it again for a week." Can you change the design of your system so that maintenance doesn't have to take the machine away to run diagnostics? "Oh, I can't give this report to Mike looking like this. He runs this whole place—I'll put it in my word processor and make it look really good." If the reports that your product creates are given to management, you can make them high-quality presentations.

A valuable system reduces friction and irritation in the workplace

Making Culture Tangible

The cultural model (Figures 6.5 and 6.6) provides a tangible representation for these intangible forces. In a cultural model we represent *influencers* (people, organizations, and groups) in the customer's culture, showing how they influence each other. Influencers are shown as large bubbles. Because culture is felt as a weight or pressure influencing actions, the bubbles sit on one another, showing how one organization forces another to take or not take actions. We represent *influences* as arrows piercing the bubbles and label the arrows to represent the type of influence. Influences are labeled with language representing the experience of the people doing the work, so the influence from an internal help organization might read, "We are unreliable and will wipe your hard drive on a whim." No one in that help organization would ever actually *say* those words, of course, but the people who use their services operate as though they were saying exactly that. Using direct language on the model makes the culture it represents stand out. Where an influence stands out as being particularly harmful and counterproductive, we mark it with a lightning bolt, our universal symbol for problems or breakdowns.

The cultural model speaks the words people think but don't say

Cultural models do not map to organization charts. They show how power is experienced by people, rather than the formal power of the organization. So it's unusual to see the whole management chain represented on a cultural model. Individual managers will appear when they are part of the work, as when a manager makes his secretary interact with the computer for him. There's often a bubble to represent the organization's culture, with

An organization's culture is not reflected in its organization chart

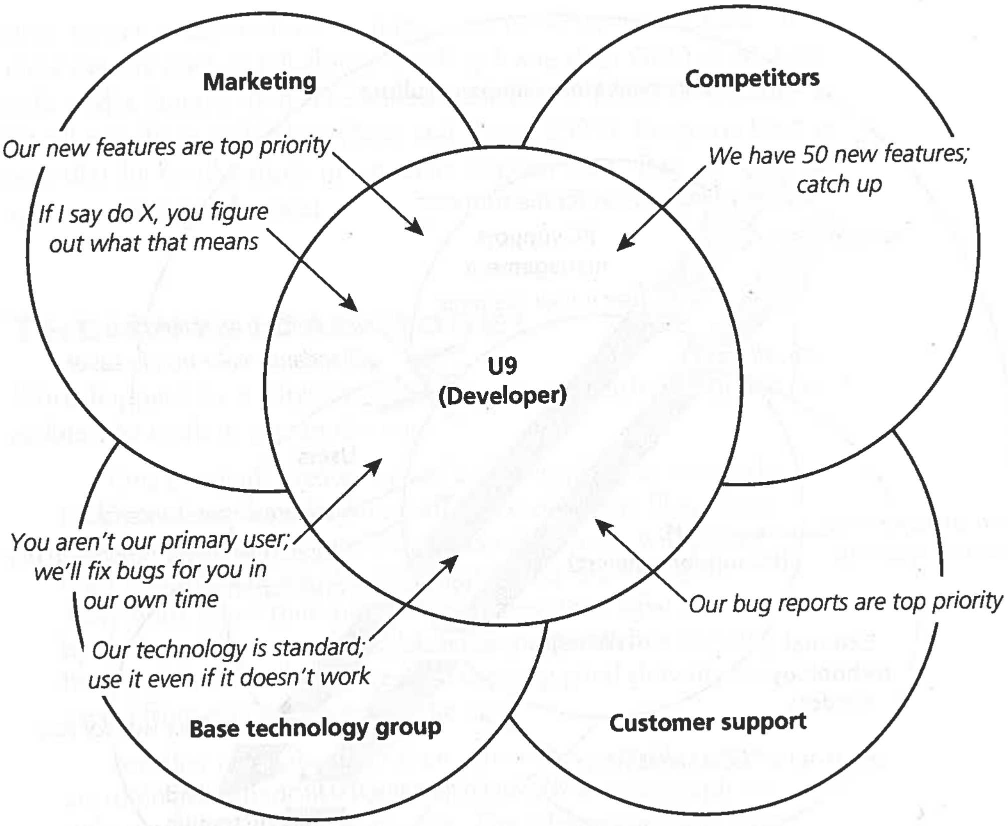

Marketing
Competitors
Our new features are top priority
If I say do X, you figure out what that means
We have 50 new features; catch up
U9 (Developer)
You aren't our primary user; we'll fix bugs for you in our own time
Our technology is standard; use it even if it doesn't work
Our bug reports are top priority
Base technology group
Customer support

FIGURE 6.5 The culture of a product development organization. This is a typical cultural model in a product development organization. In the center we see the interviewee, U9. Since cultural models are initially built as the result of an interview with one person, they represent the point of view of that one person. U9 is in the development organization, and the model shows two major constraints on them. The marketing organization constrains them through ill-specified product requirements. Competitors constrain them by creating a climate in which keeping up with the number of features is the primary goal. The basic appearance of this model—the interviewee surrounded by influencers—is very typical.

influences like "We are totally customer-focused" or "Spending money is not a problem." In adversarial situations, "management" may appear to represent how "they" do things to "us"—"We think you salesmen are children who need to be watched every moment" might be an example. Individual managers appear as managers only when they are charismatic figures who define the organization's culture. In this case,

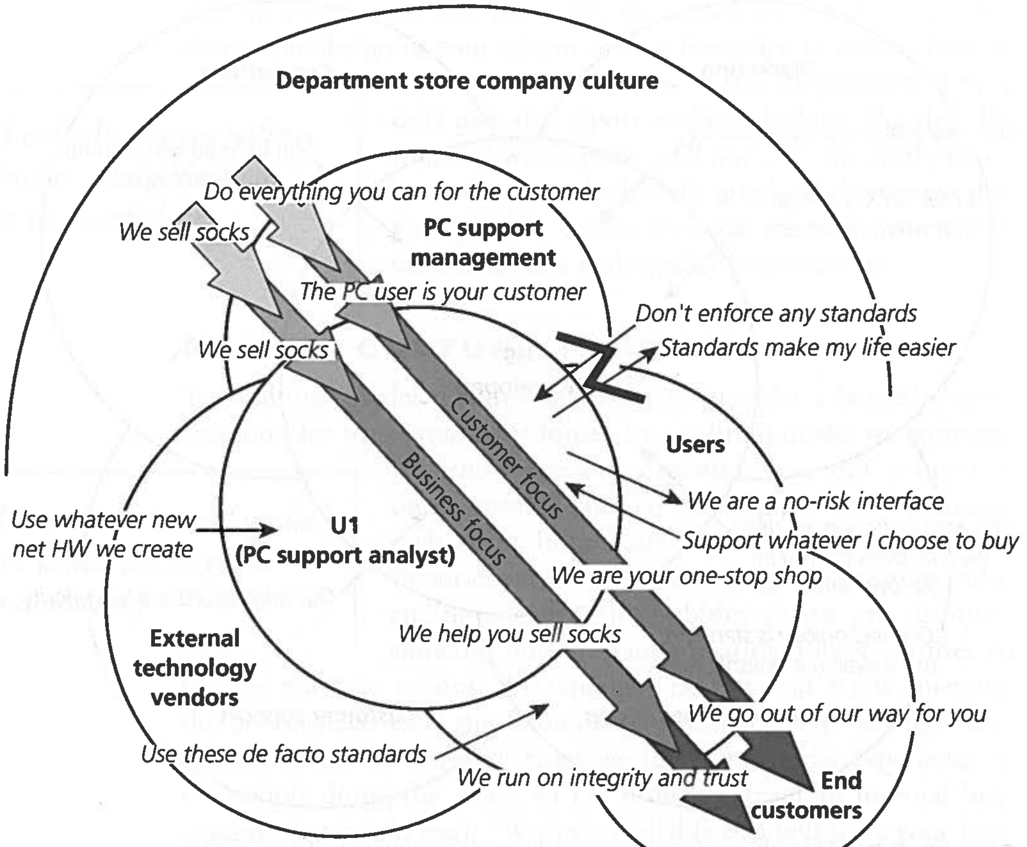

Department store company culture
Do everything you can for the customer
We sell socks
PC support management
The PC user is your customer
Don't enforce any standards
Standards make my life easier
We sell socks
Customer focus
Business focus
Users
We are a no-risk interface
Support whatever I choose to buy
Use whatever new net HW we create
U1 (PC support analyst)
We are your one-stop shop
External technology vendors
We help you sell socks
We go out of our way for you
Use these de facto standards
We run on integrity and trust
End customers

FIGURE 6.6 The culture of a customer-centered organization. This cultural model is typical when there is a definite corporate culture to account for. This cultural model represents a department store that has made customer satisfaction its first priority. Unlike many places that espouse that goal, this company has really done it—so much so that people throughout the organization are conditioned to think who their customer is and how to give them the best service. Paired with this focus on the customer is an equally pervasive understanding of the business—so much so that "We sell socks" is a watchword within the company. We show the pervasive company culture as an umbrella over everything, with individual influences going back and forth between the bubbles. The lightning bolt indicates a place where two values conflict: being customer-centered leads the store to avoid setting standards for computer configurations, but such standards would make the PC support analyst's life easier.

their power is experienced as direct and personal. McDonald's franchise owners used to tell about how Ray Kroc, then CEO of McDonald's and a fanatic about cleanliness, canceled a franchise because he found one fly in its kitchen (Boas and Chain 1977). Everyone lived in fear that he would show up in their kitchen next. Ray Kroc would appear on a cultural model.

THE PHYSICAL MODEL

Work happens in a physical environment that either supports and enables the work or gets in the way:

> One company creates a page design product in which the look on-screen doesn't quite match that of paper. They think it is close enough because they expect their users will print draft versions and use the paper output for the final draft. They don't know that most of their users don't have printers by their desks, or even close by. So users spend time running back and forth to the printer and copying good drawing elements from one document to the next.
>
> Another company gives their sales force portable computers to do presentations. They don't know that salespeople are only given a few minutes at a site. The salespeople don't have time to bring up a computer, and they depend on leaving materials behind with their customers. The portable computer doesn't fit either need.
>
> A utility company gives their electricians documentation in a three-ring binder. Only later do they discover the electricians are trying to balance this awkward binder on a cherry picker in all types of weather. They redesign the documentation as a small, spiral-bound flip book with laminated pages and a clip so it can be hung from a belt.

Any product or system must live with the constraints of the physical environment as it exists. If it ignores those constraints, it creates problems for its users. In each of the above examples, a system created problems for its users because it assumed things about the workplace that were not true. Studying the users' workplace ensures that the system accounts for the physical environment.

The physical model reveals design constraints

The physical environment constrains what people can do, but within those constraints people do have some control over their environment. Studying the workplace offers important clues to the way people structure and think about work. People restructure their workplace to support doing work in the way they prefer, to the extent they can. Because they structure their environment to be convenient, the structures they create mirror their thought. The structures show what people group together into conceptual units and coherent tasks. An office worker sets up places in her office to keep her work organized. The chair receives urgent messages from coworkers; the space next to the computer is kept clear so that when she starts a task, she has a place to lay it out; the in-box is the "guilt pile"—things she feels that she ought to deal with, when she has time. The places she creates mirror the way she thinks about her work: urgent, current, guilt pile. They make work distinctions concrete. A system that makes these distinctions real will fit with the work easily. The workplace shows us issues in doing work; from the elaborate system of piles that people create, we can deduce that tracking multiple little tasks is a problem, and people might benefit from better ways to track them.

Model both site and workplace

The physical environment is the world people live in: the rooms, cars, buildings, and highways they move about and work in; how each of these spaces is laid out so that it supports work; and how they use these spaces in the process of working. It includes how they move about, how the space supports or hinders communication, and the location of the tools people use (hardware, software, networks, machines) to do work. The physical environment affects how work is done at every scale: the multiple sites and their relationships to each other, the structure of a single site, and an individual's workplace. The work site may be structured as an open "bull pen" with supervisors' offices around the outside. It may consist of many individual cubicles dividing up a large room. A person's workplace may be an entire building or buildings, if they are maintaining equipment. It may be a car or airplane if they work on the road. Within a work site there are places to do work, which may be offices, labs, workbenches, or workstations. Workstations may be dedicated to one person or shared.

People reorganize their environment to reflect the work they do

PHYSICAL MODEL DISTINCTIONS

The *places* in which work occurs: rooms, workstations, offices, and coffee stations. The model shows whether the space is small or large, a primary or secondary workplace, private or open, cluttered, or empty space available for changing work activities.

The physical *structures* that limit and define the space: sites, walls, basements, desks, file cabinets, and other large objects.

The *usage* and *movement* within the space—how people move about in it and move things about in it in the course of accomplishing their work.

The *hardware, software, communication lines,* and other *tools* (calculator, Rolodex, in-basket, measuring tools, Post-its, printer, fax) that are present in the space and support the work or seem related. We show network connections, not to model the network itself, but to emphasize who is connected to whom and therefore what communication among people we can automate.

The *artifacts* that people create, modify, and pass around in support of the work—folders, spreadsheets, to-do lists, bills, ID cards, approvals, piles of stuff. The physical model shows the artifact and its location, not the detailed structure and usage of the artifact.

The *layout* of the tools, artifacts, movable furniture, and walls in relationship to each other to support specific work strategies.

Breakdowns or problems showing how the physical environment interferes in the work, represented as a red lightning bolt (black in this book). ❑

SEEING THE IMPACT OF THE PHYSICAL ENVIRONMENT

The physical environment is easy to see—it's all right there. It's harder to tell what matters. What will affect the design problem, and what will not? Here are some things to look for.

ORGANIZATION OF SPACE. Are there stations, and how do they relate to the work? Are stations grouped to follow the flow of work to make work efficient, or are similar stations placed together to make management efficient? Are the people who made the decision conscious of the trade-off? This will indicate what they care most about and therefore what the most important problems for you to solve are.

Planned space reflects organizational assumptions

DIVISION OF SPACE. Where are the walls, and how do they break up the work? Do they follow the structure of the work, or do they interfere with it? If they interfere, how do people overcome them? Do they run back and forth a lot? Do they shout? (During one interview, the user directed a question at the wall, and the wall answered. It was so thin he could carry on a conversation with his colleague on the other side.) Every communication breakdown creates an opportunity for you to ameliorate it: Who needs to communicate? How and when? Can you obviate the need by providing information where it's required, or can you make it easier?

Look at how people ignore walls or create walls that aren't there

GROUPING OF PEOPLE. How are people grouped into the spaces? By function or by project? Does each person have their own separate office area, or do they mix and share spaces? Often specialists sit with other specialists, not with the project they are assigned to. Creating a sense of belonging to the project team becomes difficult. Conversely, developers who are seated with their internal clients tend to identify with them. They tend to adopt their perspective against that of the development organization. What can you do to make the whole interrelated set of information systems apparent to all developers, so they are continually reminded of the effect their short-term fixes will have on the whole?

Find barriers to community and communication

ORGANIZATION OF WORKPLACES. How are the individual stations, offices, or work areas organized? How do they support the work? What is kept out (immediately visible), ready to hand (accessible without moving), and available (in a drawer or across the office)? What does this say about what's most important to the work? Things kept together tend to be used together. What does this say about the structure of a task? Can you see what makes up a whole task in what is kept together for easy access? Can you design your system so that the most important function is available where needed and so that whole tasks are coherent in the system?

Placement of objects and piles makes the work efficient

MOVEMENT. When do people move? Why do they leave one place and go to another? What triggers them to do so? Is this intrinsic to the work, as when a maintenance person goes to look at a machine? Does it provide an opportunity for informal discussion and problem solving? Do the customers see it as a problem, or are they like system support people, who generally enjoy getting out of their offices? Understanding why the movement happens will help you decide whether it makes more sense to support it better or eliminate it.

Movement reveals human preference and work needs

SHOWING WHAT MATTERS IN THE PHYSICAL ENVIRONMENT

A physical model (Figures 6.7 and 6.8) is a drawing of those aspects of the workplace that are related to the project focus. The physical model shows how the physical environment affects the work. It is annotated to show how the space is used and to show strategies, intents, and cultural values that are revealed by the way space is used. A good physical model evokes the experience of the workplace in the same way as a caricature. Aspects of the environment are only represented if they matter to the work; for example, "basement" might mean "far away, uncomfortable, and inconvenient to get to." If the worker must nonetheless go there or worry about what happens there, we represent it in the model. Wherever the physical environment interferes with the customer's work—things are too far away, or too cramped, or the right tools aren't where they are needed—we show it with a lightning bolt.

The physical model is a caricature of the workplace, not a floor plan

The physical model shows how people respond to the environment by restructuring it. Do people accept the workplace as it is, or do they work around it? If the environment consists of doorless cubicles, do they put things in front of the door to gain a measure of privacy? How else does the work as it is experienced mismatch work as the environment wants it to be? What do people do about it?

A physical model is not a floor plan for the work site. Nor is it an inventory of the computer room. Either a floor plan or an inventory can be collected easily, without resorting to contextual techniques. A physical model does not show extraneous detail unrelated to the project focus—potted plants, kids' toys, and family pictures are usually not relevant and can be omitted when you're designing a system. If you were designing the work environment itself, you might have to take them into account.

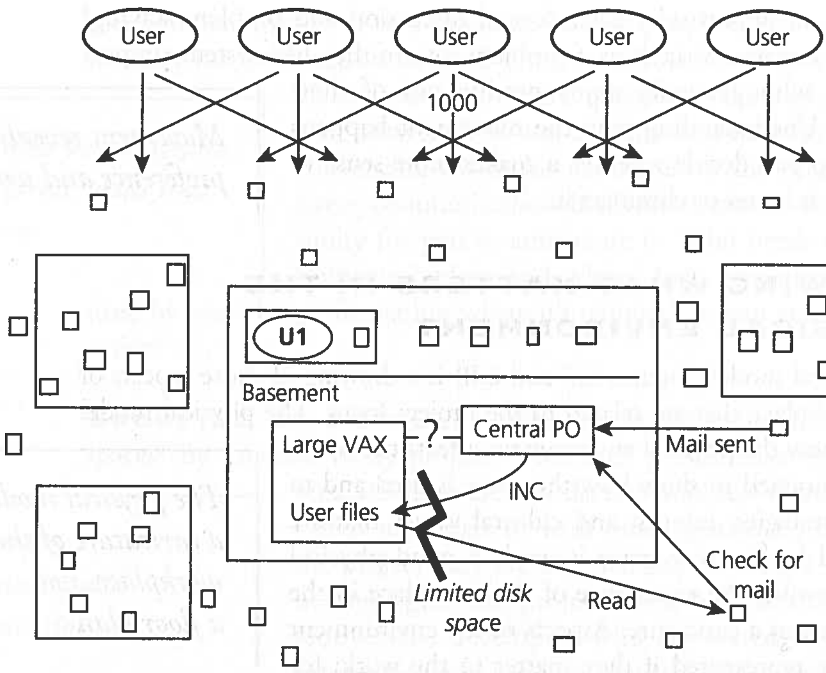


FIGURE 6.7 Physical model for a university environment. This university has set up its workstations so that anybody can use any workstation. The small boxes represent the workstations—over 1000 distributed all around the campus. To indicate their independent nature, we show them as standing alone and show the users separated from them, to indicate any user can access any workstation. "U1" means "user 1" and indicates the office of the user we talked to in a central building, with the central VAX machines in its basement. All user files are stored on the VAX. The "central PO" is a piece of software that routes mail between users. We have shown the routing of one message because we were designing a communications product. This work model shows the value of choosing a representation that is expressive of the data—in this case, that there are many workstations spread out over the campus according to no particular plan.

THE FIVE FACES OF WORK

Each of the above work models presents a different perspective on the work. These perspectives interlock: a person plays roles; a role has responsibilities, undertakes tasks, and exchanges artifacts with other people to discharge these responsibilities. The sequence models show how these tasks are accomplished in detail and how artifacts are used in accomplishing them. The responsibilities and manner of accomplishing

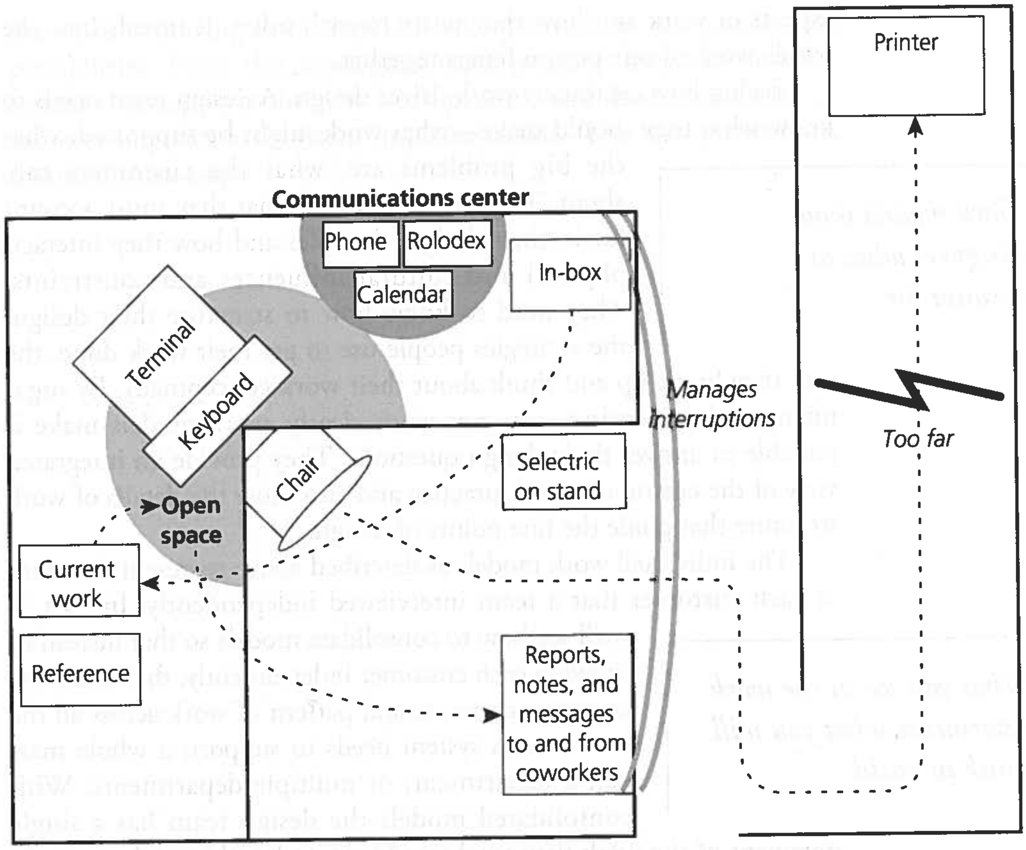


FIGURE 6.8 Physical model for an office. This physical model shows the workplace of one user. The model represents a cubicle and shows how she has structured her environment to help her get work done. The placement of her IBM Selectric in the doorway, the in-box next to the door, and the shelf used as a drop-off place all suggest a strategy to minimize interruptions caused by working in an open cubicle. The phone, Rolodex, and calendar are all grouped together, suggesting that these tools work together to support communicating and coordinating with others. And the open space around her workstation suggests an intent to keep this area clear so she can lay out her next task. The team has annotated the model to reveal these distinctions and to show breakdowns, such as the printer being too far.

them are driven by organizational context and culture as shown on the cultural model. The work represented by the sequences is done within the work environment described by the physical model. Stepping back and looking at the models together reveals all the different aspects of work and how they relate to each other. It reveals how the whole work of one person hangs together.

Seeing how customers work drives design. A design team needs to know what they should make—what work might be supported, what the big problems are, what the customers care about. They need to know what they must account for in their design: the roles and how they interact, physical and cultural influences and constraints. They need to know how to structure their design: the strategies people use to get their work done, the way they break up and think about their work conceptually. By organizing and presenting customer work clearly, work models make it possible to answer these design questions. They provide an integrated view of the customer's work practice and also show the details of work structure that guide the fine points of design.

Work models show designers what to account for

The individual work models as described above represent the work of each customer that a team interviewed independently. In Part 3, we'll see how to consolidate models so that instead of showing each customer independently, they show the common structure and pattern of work across all the customers a system needs to support: a whole market, a department, or multiple departments. With consolidated models the design team has a single statement of the work they need to address, rather than trying to support each individual separately. We do this by first observing, inquiring into, and representing the work of specific individuals. Then we consolidate the models of each type. We bring all individual flow models together into one consolidated flow model to reveal the common roles and their interaction. We consolidate all the cultural models, all the physical models of whole sites, and all the physical models of individual workplaces. We consolidate all the sequences representing similar tasks and all the artifacts achieving the same intent.

What you see in the work determines what you will think to build

These consolidated models make the underlying patterns of work across customers explicit. At the same time, they capture the variation between customers by showing any unique structure or details put into practice by each customer site. The design team can then decide what aspects of work they want to support. They can take a good idea for approaching the work implemented by one customer site and build it into the system to make it available to all. They can streamline the work, removing extra steps and taking advantage of technological possibilities. From this redesigned work practice, they can design a system that supports the new work practice and drives the design of the user interface and system implementation. The rest of this book discusses these steps.