Chapter 12

Using computers: A direction for design

This book is concerned with the design of computer-based systems to facilitate human work and interaction. In this final chapter we suggest directions for the future, drawing on the discourse developed in Part I to re-examine some basic questions about what designing means.

The most important designing is *ontological.*[1] It constitutes an intervention in the background of our heritage, growing out of our already-existent ways of being in the world, and deeply affecting the kinds of beings that we are. In creating new artifacts, equipment, buildings, and organizational structures, it attempts to specify in advance how and where breakdowns will show up in our everyday practices and in the tools we use, opening up new spaces in which we can work and play. Ontologically oriented design is therefore necessarily both reflective and political, looking backwards to the tradition that has formed us but also forwards to as-yet-uncreated transformations of our lives together. Through the emergence of new tools, we come to a changing awareness of human nature and human action, which in turn leads to new technological development. The designing process is part of this 'dance' in which our structure of possibilities is generated.

The concluding sections of this chapter will discuss the ontical-ontological significance of design—how our tools are part of the background in which we can ask what it is to be human. First we consider the direct relevance of our theoretical orientation to the design of computer systems. We

[1] We do not use 'design' here in the narrow sense of a specific methodology for creating artifacts, but are concerned with a broad theory of design like that sought in the work of reflective architects such as Alexander (*Notes on the Synthesis of Form,* 1964).

will use conversation systems, like the coordinator described in Chapter 11, as our primary example. But our intended scope is larger, encompassing other computer systems and ultimately all technology.

12.1 A background for computer design

In the popular literature on computers, one frequently encounters terms such as 'user-friendly,' 'easy-to-learn,' and 'self-explaining.' They are vague and perhaps overused, but they reflect real concerns—concerns that are not adequately understood within the rationalistic tradition, and to which phenomenological insights about readiness-to-hand, breakdown, and blindness are relevant.

Readiness-to-hand

One popular vision of the future is that computers will become easier to use as they become more like people. In working with people, we establish domains of conversation in which our common pre-understanding lets us communicate with a minimum of words and conscious effort. We become explicitly aware of the structure of conversation only when there is some kind of breakdown calling for corrective action. If machines could understand in the same way people do, interactions with computers would be equally transparent.

This transparency of interaction is of utmost importance in the design of tools, including computer systems, but it is not best achieved by attempting to mimic human faculties. In driving a car, the control interaction is normally transparent. You do not think "How far should I turn the steering wheel to go around that curve?" In fact, you are not even aware (unless something intrudes) of using a steering wheel. Phenomenologically, you are driving down the road, not operating controls. The long evolution of the design of automobiles has led to this readiness-to-hand. It is not achieved by having a car communicate like a person, but by providing the right coupling between the driver and action in the relevant domain (motion down the road).

In designing computer tools, the task is harder but the issues are the same. A successful word processing device lets a person operate on the words and paragraphs displayed on the screen, without being aware of formulating and giving commands. At the superficial level of 'interface design' there are many different ways to aid transparency, such as special function keys (which perform a meaningful action with a single keystroke), pointing devices (which make it possible to select an object on the screen), and menus (which offer a choice among a small set of relevant actions).

More important is the design of the domains in which the actions are generated and interpreted. A bad design forces the user to deal with complexities that belong to the wrong domain. For example, consider the user of an electronic mail system who tries to send a message and is confronted with an 'error message' saying "Mailbox server is reloading." The user operates in a domain constituted of people and messages sent among them. This domain includes actions (such as sending a message and examining mail) that in turn generate possible breakdowns (such as the inability to send a message). Mailbox servers, although they may be a critical part of the implementation, are an intrusion from another domain—one that is the province of the system designers and engineers. In this simple example, we could produce a different error message, such as "Cannot send message to that user. Please try again after five minutes." Successful system builders learn to consider the user's domain of understanding after seeing the frustrations of people who use their programs.

But there is a more systematic principle at stake here. The programmer designs the language that creates the world in which the user operates. This language can be 'ontologically clean' or it can be a jumble of related domains. A clearly and consciously organized ontology is the basis for the kind of simplicity that makes systems usable. When we try to understand the appeal of computers like the Apple MacIntosh (and its predecessor the Xerox Star), we see exactly the kind of readiness-to-hand and ontological simplicity we have described. Within the domains they encompass—text and graphic manipulation—the user is 'driving,' not 'commanding.' The challenge for the next generation of design is to move this same effectiveness beyond the superficial structures of words and pictures into the domains generated by what people are doing when they manipulate those structures.

Anticipation of breakdown

Our study of Heidegger revealed the central role of breakdown in human understanding. A breakdown is not a negative situation to be avoided, but a situation of non-obviousness, in which the recognition that something is missing leads to unconcealing (generating through our declarations) some aspect of the network of tools that we are engaged in using. A breakdown reveals the nexus of relations necessary for us to accomplish our task. This creates a clear objective for design—to anticipate the forms of breakdown and provide a space of possibilities for action when they occur. It is impossible to completely avoid breakdowns by means of design. What can be designed are aids for those who live in a particular domain of breakdowns. These aids include training, to develop the appropriate understanding of the domain in which the breakdowns occur and also to develop the skills and procedures needed to recognize what has broken down and how to cope with the situation.

Computer tools can aid in the anticipation and correction of breakdowns that are not themselves computer breakdowns but are in the application domain. The commitment monitoring facilities in a coordinator are an example of such a system, applied to the domain of conversations for action. In the design of decision support systems, a primary consideration is the anticipation of potential breakdowns. An early example of such a system was Cybersyn,[2] which was used for monitoring production in a sector of the economy of Chile. This system enabled local groups to describe the range of normal behavior of economic variables (such as the output of a particular factory), and to be informed of significant patterns of variation that could signal potential breakdown.

But more importantly, breakdowns play a fundamental role in design. As the last section pointed out, the objects and properties that constitute the domain of action for a person are those that emerge in breakdown. Returning to our simple example of an electronic mail system, our 'fix' left a person with certain courses of action in face of the breakdown. He or she can simply forget about sending the message or can wait until later to try sending it again. But it may be possible to send it to a different 'mail server' for delayed forwarding and delivery. If so, it is necessary to create a domain that includes the existence of mail servers and their properties as part of the relevant space in which the user exists.

In designing computer systems and the domains they generate, we must anticipate the range of occurrences that go outside the normal functioning and provide means both to understand them and to act. This is the basis for a heuristic methodology that is often followed by good programmers ("In writing the program try to think of everything that could go wrong"), but again it is more than a vague aphorism. The analysis of a human context of activity can begin with an analysis of the domains of breakdown, and that can in turn be used to generate the objects, properties, and actions that make up the domain.

The blindness created by design

Any opening of new possibilities closes others, and this is especially true of the introduction of technology. As an example, consider the possibility of an 'electronic library' in which one can search for items using sophisticated cataloging techniques based on publication information (such as author, publisher, and title) and topic classifications (such as the Library of Congress categories and the key word systems used in many journals).

[2]Cybersyn is described in Beer, *Platform for Change* (1975).

If we accept the domain generated by those classifications as the relevant one for finding books, the system is appropriate and useful. However, this may not be the right choice. The facility may make it easier for a reader to find a book on a specific narrow topic, while reducing the ease of 'browsing' through shelves of loosely related material. Recognizing the importance of background and thrownness, it becomes clear that the unexpected and unintended encounters one has in browsing can at times be of much greater importance than efficient precise recall. If the problem is narrowly construed as "Find a book, given specific information" then the system may be good. If we put it into its larger context of "Find writings relevant to what you want to do" it may well not be, since relevance cannot be formalized so easily. In providing a tool, we will change the nature of how people use the library and the materials within it.

As with breakdown, blindness is not something that can be avoided, but it is something of which we can be aware. The designer is engaged in a conversation for possibilities. Attention to the possibilities being eliminated must be in a constant interplay with expectations for the new possibilities being created.

12.2 A design example

We turn now to a concrete example of how our theoretical background might guide the design of a computer-based system in a practical setting. It is not a complete analysis of the specific case, but is a vehicle for suggesting possibilities and clarifying the points in these two concluding chapters. We have chosen a mundane business example, but the same principles hold for applications of computers in all kinds of organizations.

> *The setting:* You have been operating a successful dress shop for several years and expanded last year to a chain of three stores. You have not made any use of computers, but have recently come to feel they might be of some help. Profits aren't what they should be, you are losing some customers who seem dissatisfied with the service they get, and the staff feels overworked.

There are no clear problems to be solved: Action needs to be taken in a situation of irresolution.

This is the typical case in which questions about what to do arise, as described in Chapter 11. There is no clear 'problem' to be solved, but a sense of irresolution that opens opportunities for action. Computers are not the 'solution,' but may be useful in taking actions that improve the situation. Once the manager senses this, the typical next step would be to go to computer service vendors to find out what kinds of 'systems' are available and to see if they are worth getting. The space of possibilities is determined by the particular offerings and the 'features' they exhibit. But we can begin with a more radical analysis of what goes on in the store and what kinds of tools are possible.

A business (like any organization) is constituted as a network of recurrent conversations.

As a first step we look for the basic networks of conversation that constitute the business. We ask "Who makes requests and promises to whom, and how are those conversations carried to completion?" At a first level we treat the company as a unity, examining its conversations with the outside world—customers, suppliers, and providers of services. There are some obvious central conversations with customers and suppliers, opened by a request for (or offer of) dresses in exchange for money. Secondary conversations deal with conditions of satisfaction for the initial ones: conversations about alteration of dresses, conversations concerning payment (billing, prepayment, credit, etc.), and conversations for preventing breakdown in the physical setting (janitorial services, display preparation, etc.).

Taking the business as a composite, we can further examine the conversational networks among its constituents: departments and individual workers. There are conversations between clerk and stockroom, clerk and accounting, stockroom and purchasing, and so forth. Each of these conversation types has its own recurrent structure, and plays some role in maintaining the basic conversations of the company. As one simple example, consider the conversation in which the stock clerk requests that purchase orders be sent to suppliers. Instances of this conversation are either triggered by a conversation in which a salesperson requested an item that was unavailable, or when the stock clerk anticipates the possibility of such a breakdown. Other conversations are part of the underlying structure that makes possible the participation of individuals in the network (payroll, work scheduling, performance evaluations, etc.). Each conversation has its own structure of completion and states of incompletion with associated time constraints.

Conversations are linked in regular patterns of triggering and breakdown.

The goal in analyzing these conversations is a description in which the linkages between the recurrent conversations are made explicit. These links include normal triggering (e.g., a customer request triggers a stock request), and others that deal with breakdown (e.g., if a request for alteration is not met on time it may trigger a request by the customer to see the manager). Having compiled this description, we can see possibilities for restructuring the network on the basis of where conversations fail to be completed satisfactorily. We may, for example, note that customer dissatisfaction has come from alterations not being done on time (perhaps because alterations are now being combined for the three stores and therefore the tailors aren't immediately available). Actions might include imposing a rigid schedule for alterations (e.g., never promise anything for less than a week) so that commitments will be met on time, even if the times that can be promised are less flexible. Or it might mean introducing better tools for coordination, such as a computer-based system for keeping track of alteration requests and giving more urgent ones higher priority.

In creating tools we are designing new conversations and connections.

When a change is made, the most significant innovation is the modification of the conversation structure, not the mechanical means by which the conversation is carried out (e.g., a computer system versus a manual one based on forms). In making such changes we alter the overall pattern of conversation, introducing new possibilities or better anticipating breakdowns in the previously existing ones. This is often not noticed because the changes of devices and of conversation structure go hand in hand. At times the changes can be beneficial, and at times detrimental. There are many cases of systems for activities like job scheduling that were introduced to make things more efficient, but as a result of modifying the conversation structure they in fact hindered the work. Often this is the result of taking one part of the conversation network (the 'official' or 'standard' part) and embodying it in the structure of the computer system, thereby making impossible other less frequent types of requests and promises that are crucial for anticipating and coping with breakdowns. When we are aware of the real impact of design we can more consciously design conversation structures that work.

As an example, there is a potential for coordination systems to reduce the need for rigid work schedules. Much of the temporal structure of what goes on in organizations is driven by the need to be able to anticipate completion. If the manager knows that a certain task will be done every Friday, then he or she can make a commitment to do something that uses its results on the following Monday. For many routine tasks, this is the best way to guarantee effective coordination. But it can also be an inflexible straitjacket that reduces the space of possibilities open to workers in organizing their activities. If effective coordination on a conversation-by-conversation basis could be regularized, then the rigidity could be relaxed, altering the conversation structure to make the workers more productive.

Design includes the generation of new possibilities.

No analysis of existing recurrent structures is a full account of the possibilities. The existing networks represent a particular point of structural coupling of the organization to the world in which it exists. Actions may radically alter the structure. In our example, the store might simply stop doing alterations. Or it might hire more tailors, or contract out the alterations, or hire many more tailors and go into the contract alteration business as well. In some cases, the business as a whole may have a new interpretation. The owner of a small candy store notes the success of the video games in the back, and may ultimately decide that the business is a video game parlor with a candy counter. No methodology can guarantee that all such possibilities will be found, but a careful analysis of the conversation structure can help reveal conversations with a potential for expansion.

In designing computer-based devices, we are not in the position of creating a formal 'system' that covers the functioning of the organization and the people within it. When this is attempted, the resulting system (and the space of potential action for the people within it) is inflexible and unable to cope with new breakdowns or potentials. Instead we design additions and changes to the network of equipment (some of it computer-based) within which people work. The computer is like a tool, in that it is brought up for use by people engaged in some domain of action. The use of the tool shapes the potential for what those actions are and how they are conducted. The computer is unlike common tools in its connectivity to a larger network of equipment. Its power does not lie in having a single purpose, like a carpenter's plane, but in its connection to the larger network of communication (electronic, telephone, and paper-based) in which organizations operate.

Domains are generated by the space of potential breakdown of action.

If our dress shop owner chooses to install a computer-based system dealing with some of the conversations, the analysis proceeds by examining (and generating) the appropriate domains. Much computer automation deals with standard derived domains, such as payroll accounting, billing, and employee scheduling. A domain of relevant objects, properties, and actions has already been generated through standard practice, and is enforced by the need to satisfy external conversations based on it (such as those with the Internal Revenue Service). But even in these sedimented cases, it is important to recognize that ultimately the present-at-hand world of objects is always based on the breakdown of action.

As an obvious example, we can ask what a customer's 'address' is. The immediate response is "For what?" (or, "What is the conversation in which it determines a condition of satisfaction?"). There are two distinct answers. Some conversations with customers involve requests for the physical transfer of goods while others involve correspondence. Different conditions of satisfaction require different kinds of address. This is a standard case, and most business forms and computer data bases will distinguish "shipping address" and "billing address." But we may also need an address where the person can be found during the day to perform further measurements. In every case, the relevant 'property' to be associated with the person is determined by the role it plays in an action. This grounding of description in action pervades all attempts to formalize the world into a linguistic structure of objects, properties, and events.

This also leads us to the recognition that the development of any computer-based system will have to proceed in a cycle from design to experience and back again. It is impossible to anticipate all of the relevant breakdowns and their domains. They emerge gradually in practice. System development methodologies need to take this as a fundamental condition of generating the relevant domains, and to facilitate it through techniques such as building prototypes early in the design process and applying them in situations as close as possible to those in which they will eventually be used.

Breakdown is an interpretation—everything exists as interpretation within a background.

As a somewhat more interesting example of how the world is generated by language, consider the conditions of satisfaction associated with 'fit.' The customer is only satisfied by a dress that fits, and a complex linguistic domain (the domain of clothing sizes) has been generated to provide a means of anticipating and preventing breakdown. But 'fitting' cannot be objectively defined. One person may be happy with an article that someone else of the same overall shape and size would reject. The history of fashion and the differences between cultures make it clear that 'fitting' is an interpretation within a particular horizon. But at the same time it is not purely individual. The background shared by a community is what makes individual 'tastes' possible.

Ultimately, then, satisfaction is determined not by the world but by a declaration on the part of the requestor that a condition is satisfied. The case of 'fit' may seem extreme, but every condition of satisfaction ultimately rests on a declaration by an individual, within the background of a community. The cases that seem 'objective' are those in which there is great regularity and for which possible conversations about satisfaction have been regularized (perhaps formally in the legal system). One kind of innovation lies in generating new interpretations and corresponding new domains for conditions of satisfaction. In fact, one might view this as the primary enterprise of the 'fashion' industry (and of every entrepreneur).

Domains of anticipation are incomplete.

The domain of clothing sizes was generated to anticipate breakdown in the satisfaction of conversations in which clothing is sold. It is a useful but incomplete attempt. Given the interpretive nature of 'fit,' no system of sizes can guarantee success. Once again, this is a clearly visible example of a more universal phenomenon. Every attempt to anticipate breakdown reflects a particular domain of anticipation. This does not make it useless, but means that we must design with the flexibility to encounter other (always unanticipated) breakdowns.

As another case, consider inventory control. The stock clerk tries to maintain a supply on hand that will decrease the possibility of running out, while keeping the overall investment in inventory as low as feasible (thereby anticipating breakdowns in cash flow). Orders are sent far enough ahead of time to anticipate delivery lags, and counts of what has been sold are used to keep track of what is on hand. But of course there are breakdowns in all of this. A supplier can simply fail to deliver as promised. An inventory count based on previous inventory and on what has been sold fails to account for the items lost through shoplifting. This does not mean that anticipation is impossible or that systems should not be built to do it. The critical part is to recognize clearly what the real domains are. An inventory count is not a statement of fact, but a declaration of an interpretation. For many purposes this can be treated as though it were the 'actual number of items,' but conversations that depend on this assumption will fail to deal with the unexpected cases.

Computers are a tool for conducting the network of conversations.

Most of what has been said in this section is independent of computers. It applies to businesses and organizations, whether they operate with the most modern equipment or with ledger pads and quills. It is also not a prescription for what they should do, but an analysis of what they are already doing. If we examine what computers are doing now in settings like our example, we find them embodying possibilities for action within a set of recurrent conversations. Whether it be a payroll system, a billing system, or an inventory control system, the hardware and software are a medium in which requests and promises are made and monitored. There is a wide range of possibilities, including the standard packages now prominent in commercial applications, the decision support systems and coordinators described in Chapter 11, and the 'expert' systems being widely promoted today. In each case, the question to be asked is not an abstract one of "What kind of system is needed?" but a concrete one of how different tools will lead to new conversations and in the end to new ways of working and Being. 'Computerization' in its pejorative sense occurs with devices that were designed without appropriate consideration of the conversational structures they engender (and those that they consequently preclude).

Innovations have their own domains of breakdown.

We have not tried to deal in our dress shop example with concrete questions of computer devices. In practice one needs to make many choices based on the availability, utility, and cost of different kinds of equipment—computers, software packages, networks, printers, and so on. In doing so, all of the same theoretical considerations apply. As computer users know all too well, breakdown is a fundamental concern. It is important to recognize in this area that breakdowns must be understood within a larger network of conversation as well. The issue is not just whether the machine will stop working, but whether there is a sufficient network of auxiliary conversations about system availability, support, training, modification, and so on. Most of the well-publicized failures of large computer systems have not been caused by simple breakdowns in their functioning, but by breakdowns in this larger 'web of computing'[3] in which the equipment resides.

Design is always already happening.

Imagine yourself in the situation depicted at the beginning of the section. You resolve to take actions that will lead to acquiring and installing a new computer system. What does our analysis have to offer? Aren't the available computer systems good enough? What guidance is there in determining what to do or buy?

Our first response is that we are not proposing some new answer to the 'data processing problem.' Much of our theoretical analysis applies to existing systems, and many of these operate in ways that achieve what

[3]This term is from Kling and Scacchi, "The web of computing" (1982), which is based on empirical studies of experience with large scale computer systems in a social context.

we propose. This is not surprising, since a situation of natural selection applies—those systems that work ultimately survive.

But this is not the whole picture. It is not necessary to belabor what everyone knows from experience—computer systems are frustrating, don't really work right, and can be as much of a hindrance as a help in many situations. We don't offer a magic solution, but an orientation that leads to asking significant questions. The result of an analysis like the above might well be to lead the shop owner to make changes to the conversations as they now occur (by voice and writing) without buying a computer at all. Or it might serve as a background from which to generate criteria for deciding among competing vendors and creating new interpretations for the available systems within the particular situation. Or it might be the basis for coming up with entirely new tools that open new possibilities for action. Design always proceeds, with or without an articulated theory, but we can work to improve its course and its results.

12.3 Systematic domains

The previous sections point out the central role played by the creation through language of the domains in which we act. Language is the creation of distinctions: nouns distinguish objects into groups, verbs distinguish kinds of actions, etc. This is not something we choose to do, but is a fundamental condition of using language. Furthermore, the words are constitutive of the objects among which they distinguish. As we showed at length in Chapter 5, language does not describe a pre-existing world, but creates the world about which it speaks. There are whole domains, such as those in financial markets involving 'shares,' 'options,' and 'futures, whose existence is purely linguistic—based on expressions of commitment from one individual to another.

The use of a distinction is very different from its explicit formal articulation. The fact that we commonly use a word does not mean that there is an unambiguous formal way to identify the things it denotes or to determine their properties. But whenever there is a recurrent pattern of breakdown, we can choose to explicitly specify a *systematic domain*, for which definitions and rules are articulated.

The preceding chapters have repeatedly contrasted the computational manipulation of formal representations with the being-in-the-world of human thought and understanding. In each case we have shown how the projection of human capacities onto computational devices was misleading. But there is a positive side to this difference. Computers are wonderful devices for the rule-governed manipulation of formal representations, and there are many areas of human endeavor in which such manipulations are crucial. In applying computers appropriately to systematic domains, we develop effective tools.

The development of systematic domains is of course not new. Mathematics is a prototypical example of such a domain, and the development of a calculus of logical form, as begun by philosophers such as Frege and Russell, made it possible to apply mathematical techniques to more general representations of objects and their properties. Work in computer science has added a new dimension—the design of mechanisms that can carry out complex sequences of symbolic manipulations automatically, according to a fixed set of rules.

There are many domains in which such manipulations are commonplace. One of the most obvious is the numbers representing financial entities and transactions. Every accounting program, payroll program, and billing program operates within a systematic domain of bookkeeping that has evolved over centuries of commercial experience. The advent of computers has not yet had a major impact on the structure of that domain, but it has made it possible to do quickly and efficiently what was previously tedious and costly.

Nobody would argue that an accounting program like Visicalc[4] 'thinks' about business, but it is a vital tool because of the clear and appropriate correspondence between its domain and the activities that generate the commercial world. Another widespread example is 'word processing,' as illustrated in our introductory chapter. Its domain is the superficial stuff of language—letters and punctuation marks, words, sentences, and paragraphs. A word processor does not 'understand' language, but can be used to manipulate text structures that have meaning to those who create and read them. The impact comes not because the programs are 'smart' but because they let people operate effectively in a systematic domain that is relevant to human work.

We can best understand the creation of expert systems as the creation of systematic domains that are relevant and useful to a particular profession. In developing such a system, there is an initial period of 'knowledge acquisition,' during which professionals in the domain work together with 'knowledge engineers' to articulate the structure of the relevant concepts and rules. This is often described as a process of 'capturing' the knowledge that the experts already have and use. In fact, it is a creative design activity in which a systematic domain is created, covering certain aspects of the professionals' work. The successful examples of expert systems have almost all been the result of long and intensive effort by a particularly qual-

[4]Visicalc is a microcomputer program that lets a person manipulate an 'electronic spreadsheet' with rows and columns of related figures. It is one of the most commercially successful pieces of software ever created, and is credited with motivating the purchase of more small home and business computers than any other single program.

ified practitioner, and it can well be argued that the domains generated in developing the system are themselves significant research contributions.

Such *profession-oriented domains* can be the basis for computational tools that do some tasks previously done by professionals. They can also be the basis for tools that aid in communication and the cooperative accumulation of knowledge. A profession-oriented domain makes explicit aspects of the work that are relevant to computer-aided tools and can be general enough to handle a wide range of what is done within a profession, in contrast to the very specialized domains generated in the design of a particular computer system. A systematic domain is a structured formal representation that deals with things the professional already knows how to work with, providing for precise and unambiguous description and manipulation. The critical issue is its correspondence to a domain that is ready-to-hand for those who will use it.

Examples of profession-oriented systematic domains already exist. One of the reasons for Visicalc's great success is that it gives accountants transparent access to a systematic domain with which they already have a great deal of experience—the spreadsheet. They do not need to translate their actions into an unfamiliar domain such as the data structures and algorithms of a programming language. In the future we will see the development of many domains, each suited to the experience and skills of workers in a particular area, such as typography, insurance, or civil engineering.

To some extent, the content of each profession-oriented domain will be unique. But there are common elements that cross the boundaries. One of these—the role of language in coordinated action—has already been discussed at length. The computer is ultimately a *structured dynamic communication medium* that is qualitatively different from earlier media such as print and telephones. Communication is not a process of transmitting information or symbols, but one of commitment and interpretation. A human society operates through the expression of requests and promises among its members. There is a systematic domain relevant to the structure of this network of commitments, a domain of 'conversation for action' that can be represented and manipulated in the computer.

Another widely applicable domain is the specification of mechanisms like those in computer hardware and programs. These involve physically embodied systems that can be understood as carrying out discrete processes (processes that proceed in identifiable individual steps). There are kinds of objects, properties, and relations that are suited to describing them and that can be embodied in a systematic domain. Programming languages are one approach to formalizing this domain, but in general they are not well suited to the communication of intent and conceptual structure. They are too oriented to the structure of the machine, rather than to the structure of its behavior. We are beginning to see the development of 'system specification languages'[5] that deal with the domain of computational devices in a more general way.

In all situations where systematic domains are applicable, a central (and often difficult) task is to characterize the precise form and relevance of the domain within a broader orientation. In our example of coordinators, we find the embedding of a systematic domain (conversation structure) within the larger domain of language. The meaning of an utterance is not captured by a formal structure, but lies in the active listening of a hearer in a context. At the same time, its role within a particular network of requests and promises can be identified and represented in a systematic way. In a similar vein, the rows and columns of a bookkeeping program do not reflect the meaning of the economic system, but isolate one aspect that is amenable to systematic treatment. The limitations of this domain become obvious in attempts to apply accounting techniques to non-systematic areas, such as measuring overall 'productivity' or providing a cost-benefit analysis of activities (such as research) whose 'products' are not easily measured.

Even within areas such as law—where there is a primary concern with the social and ethical fabric—we find an interaction between the contextual and the systematic. The statutes and decisions provide a systematic framework that is the basis for argumentation in court. There are clear formal statements, such as "In order to be guilty of first-degree murder, there must be premeditation." But of course these rest on understandings of terms like 'premeditation,' which call for contextual interpretation. Computer programs can help a lawyer manipulate formal structures and the deductions that can be made from them, while leaving the 'hard questions' open to human interpretation.[6]

12.4 Technology and transformation

Our book has focussed on the designing of computer-based tools as part of a larger perspective of ontological design. We are concerned with what happens when new devices are created, and with how possibilities for innovation arise. There is a circularity here: the world determines what we can do and what we do determines our world. The creation of a new device or systematic domain can have far-reaching significance—it can create new ways of being that previously did not exist and a framework for actions that would not have previously made sense. As an example, systematic bookkeeping techniques did not just make it easier to keep track of the

[5]See Winograd, "Beyond programming languages" (1979).

[6]See Gardner, *An Artificial Intelligence Approach to Legal Reasoning* (in press), for an example and a general discussion of the issues.

financial details of business as it existed. New ways of doing business (in fact, whole new businesses dealing with financial transactions) became possible, and the entire financial activity of society evolved in accord with the structure of the new domain.

The hermeneutic orientation of Chapter 3 and the biological theories of Chapter 4 give us insight into this process. In the act of design we bring forth the objects and regularities in the world of our concern. We are engaged in an activity of interpretation that creates both possibilities and blindness. As we work within the domain we have defined, we are blind to the context from which it was carved and open to the new possibilities it generates. These new possibilities create a new openness for design, and the process repeats in an endless circle.

In Maturana's terms, the key to cognition is the plasticity of the cognitive system, giving it the power of structural coupling. As the domain of interactions is modified, the structure of the interacting system changes in accord with it. We cannot directly impose a new structure on any individual, but whenever we design changes to the space of interactions, we trigger changes in individual structure—changes to the horizon that is the precondition for understanding.

Computers have a particularly powerful impact, because they are machines for acting in language. In using them we engage in a discourse generated within the distinctions set down by their programmers. The objects, properties, and acts we can distinguish and perform are organized according to a particular background and pre-understanding. In most cases this pre-understanding reflects the rationalistic tradition we have criticized throughout this book. It includes biases about objectivity, about the nature of 'facts' (or 'data' or 'information') and their origin, and about the role of the individual interacting with the computer.

We have argued that tools based on this pre-understanding will lead to important kinds of breakdown in their use. But there is a larger problem as well. As we work with devices whose domains of action are based on an interpretation of 'data,' 'goals,' 'operators,' and so forth, we develop patterns of language and action that reflect these assumptions. These carry over into our understanding of ourselves and the way we conduct our lives. Our criticism of descriptions of human thought as 'decision making' and language understanding as the manipulation of representations is not just a prediction that certain kinds of computer programs will fail. It reflects a deeper concern with the discourse and actions that are generated by a rationalistic interpretation of human action. Computer systems can easily reinforce this interpretation, and working with them can reinforce patterns of acting that are consistent with it.[7]

[7]This effect is described in Turkle, *The Second Self* (1984).

On the other hand, where there is a danger there is an opportunity. We can create computer systems whose use leads to better domains of interpretation. The machine can convey a kind of 'coaching' in which new possibilities for interpretation and action emerge. For example, coordinator systems grew out of research on how to train people to improve their effectiveness in working with others. This training in 'communication for action'[8] reveals for people how their language acts participate in a network of human commitments. The training does not involve computers, but rests on the development of a new linguistic domain—new distinctions and descriptions that serve as a basis for action. The coordinator can help develop and reinforce this new understanding. Even at the simple level of providing the initial possibilities of 'make request' and 'make promise' instead of 'send message,' it continually reminds one of the commitment that is the basis for language. As one works successfully in this domain, the world begins to be understood in these terms, in settings far away from the computer devices.

This is just one example of a phenomenon that is only beginning to emerge in designing computers—the domain created by a design is a domain in which people live. Computers, like every technology, are a vehicle for the transformation of tradition. We cannot choose whether to effect a transformation: as designers and users of technology we are always already engaged in that transformation, independent of our will. We cannot choose what the transformation will be: individuals cannot determine the course of a tradition. Our actions are the perturbations that trigger the changes, but the nature of those changes is not open to our prediction or control. We cannot even be fully aware of the transformation that is taking place: as carriers of a tradition we cannot be objective observers of it. Our continuing work toward revealing it is at the same time a source of concealment.

However, we can work towards unconcealment, and we can let our awareness of the potentials for transformation guide our actions in creating and applying technology. In ontological designing, we are doing more than asking what can be built. We are engaging in a philosophical discourse about the self—about what we can do and what we can be. Tools are fundamental to action, and through our actions we generate the world. The transformation we are concerned with is not a technical one, but a continuing evolution of how we understand our surroundings and ourselves—of how we continue becoming the beings that we are.

[8]The training was developed by F. Flores in conjunction with Hermenet, Inc. of San Francisco.