Guidon-Manage Revisited: A Socio-Technical Systems Approach

William J. Clancey

Journal of AI in Education **4(1):5-34, 1993**

Presented at the Conference on Intelligent Tutoring Systems, Montreal, 1992

Abstract. Until the late 1980s, ITS research proceeded in a harmonious way, with almost universal agreement within the community about the nature of human knowledge and learning. With the rise of situated cognition theories, considerable confusion has developed about theories of intelligence, when and how formal subject matter theories should be taught, and the relation of instructional technology to human interactions. Now, after several years of forming a new interdisciplinary community, methods for developing instructional programs can be articulated that emphasize developing programs that fit in the classroom and workplace. These methods place previous design processes into sharp relief and help us understand situated cognition claims about the relation of theory and practice.

Keywords. **Participatory design, computer-supported collaborative work, socio-technical systems, situated learning, technical rationality, glass box design**

1 Introduction

From 1975 through 1987, my research at the Stanford Knowledge Systems Laboratory focused on ways of using expert systems for teaching (Clancey, 1987; Buchanan & Shortliffe, 1984). We developed methods for automated explanation and student modeling that could be incorporated in case-method programs for teaching medical diagnosis. We modeled separately domain processes (i.e., the subject material), reasoning processes (e.g., how to do diagnosis), and communication processes (e.g., how to explain diagnostic strategies), enhancing the opportunity for reuse of the software (Clancey, 1992). Some of the modules we developed are:

—***Guidon-Watch*** (Richer and Clancey, 1985), for graphically displaying expert system reasoning,

—***Image*** (London and Clancey, 1982) and **Odysseus** (Wilkins and Clancey, 1988), for modeling diagnostic strategies,

—***NeoExpl*** (Hasling, et al., 1983), for explaining strategies.

One of the last programs we developed is ***Guidon-Manage*** (Rodolitz and Clancey, 1990), designed to help students reflect on reasoning processes in medical diagnosis. We developed a task language that abstracts the purpose of requests for patient data (e.g., "ask follow-up question," "test hypothesis," "ask general question") (Clancey, 1988a). We believed that this approach would enable a student to understand a more-experienced physician's behavior in the clinic, as well as to articulate what he or she doesn't know (e.g., "I know that I should now refine the disease hypothesis X, but I don't remember its subtypes"). Thus, in Guidon-Manage we focused on teaching metacognitive skills (Clancey, 1988b).

In this paper, I am concerned with the process by which we designed Guidon-Manage and how this differs from the approach we would follow today. My goal is to clarify the influence of situated cognition, the socio-technical systems approach, and participatory design on how we use ITS technology. This critique explains why I have not continued ITS tool development at IRL in the past five years, and how I am working with social scientists to define and fund research projects in a different way.

Specifically, my interest today is to relate programs like Guidon-Manage to medical life (Figure 1). In developing Guidon-Manage and related programs, we were exploring the space of what can be automated. We were funded as computer scientists to develop computer languages and modeling methods that could be applied to instruction. Our successes include the heuristic classification model of reasoning, the

task and metarule representation of reasoning strategies, and new forms of tutorial interaction in Guidon-Manage, Guidon-Debug, and so on. This exploration of "tool design space" is necessary and must continue. It constitutes what we generally call basic research, the aim of the Office of Naval Research program that sponsored this work.

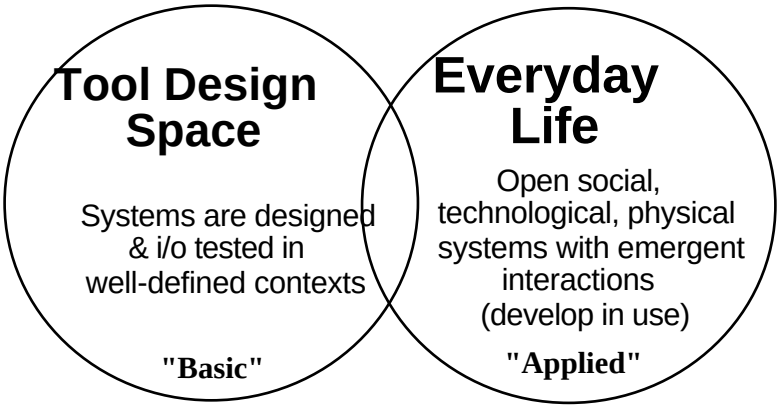

Figure 1. Contrasting research commitments

In the 1970s and 80s, we believed that our well-documented ideas would be picked up by instructional designers and applied to practical problems. Instead, we find that despite use of advanced computer technology in the 1990s, the dominant form of instructional design in schools and industry is 1960s-style page-turning presentation. No commercial authoring tool has the complexity of Guidon. At the same time, researchers in industry are finding that expert systems techniques, hatched in university laboratories, are inadequate for developing useful programs that fit into people's lives (Leonard-Barton, 1987; Mumford & MacDonald, 1989; Kukla, et al., 1990; Weitz, 1990). The problem is not just transfer of technology, but an inadequate conception of user interaction, tool requirements, and the software development process. These considerations are not exclusive to computer systems; but the difficulties are exacerbated by the profusion of control options possible in interactive, computer-controlled devices (such as a VCR), and the lack of design experience of computer programmers (Norman, 1988).

Although I could have continued to work in tool design space, I became challenged by these new problems, realizing that my original and continuing interest was to develop computer programs that would be exciting to medical students. After more than a decade, I felt that I could no longer continue saying that I was developing instructional programs for medicine, since not a single program I worked on was in

routine use (not even Mycin). This opened up the puzzle: What had I been doing and what would it mean to change medical education? I answered the first question by writing "Model Construction Operators" (Clancey, 1992a), in which I reformulated AI programming as a general method for modeling processes qualitatively. Answers to the second question came from the analysis of Brown et al. (1988) on situated learning and, unexpectedly, from extensive work on computer system design by social scientists (Zuboff, 1987; Greenbaum and Kyng, 1991).

Surprisingly, I found that applied research in software design, exemplified by the work of Greenbaum and Kyng, was basic research occurring in another setting, shifting from exploring what a computer can do to determining *through practice* what designs are useful. Complementing this view of the design process, as I will explain in more detail below, situated cognition provides guidance about what systems to build, namely *teaching abstractions in a contextualized way* (Brown, et al., 1988)[1]. We conjecture that to make systems like Guidon-Manage useful, we must help students understand where qualitative models (such as a disease taxonomy) come from, their limitations, and how to keep them up-to-date. Meeting this goal—relating an ITS program to the life of its users—requires engaging users in the design process in a way that radically changes the design process itself (Floyd, 1987; Greenbaum and Kyng, 1991). This process emphasizes incremental design in the context of use, in which every version of the program is useful and fits into the complexity of the classroom (or workplace, in the case of on-the-job training).

In short, determining *what to teach* and *how to build an ITS system* affect each other; our views about the nature of knowledge will affect not only what we teach, but how we interact with teachers. As I will explain, a socio-technical systems approach to the design process and a situated cognition view of knowledge fundamentally involve us in the practice of teachers and the practice of the subject domain. Theoretical ideas about ITS programs as well as theoretical ideas about the subject matter are brought into realignment with how artifacts and models develop in everyday life.

The difference between the two research commitments (Figure 1) can be described in terms of evaluation criteria: Exploration of what computers can do is framed in terms of well-defined contexts, with performance measured in terms of program input

[1]Note that this means teaching abstractions, such as diagnostic strategies, by relating them to everyday life. Some critics of Brown et al have wrongly interpreted situated learning to mean not teaching abstractions at all (Clancey, 1992b). The distinction is between content and how it is presented.

and output. To develop programs for everyday life, correct operation is essential, but there are more constraints, imposed by the social and physical environment. Rather than just evaluating isolated cognitive capabilities of the program (e.g., how well the explanation system works); we have instead a larger problem of *designing new socio-technical systems*. Design becomes an integrative process with many competing voices, concerning the social, physical, and information-processing environments. *The design process* becomes our research focus.

Crucially, we engage in incremental development and evaluation because design ideas, how the program is used, and its value to people will emerge through use, an idea described by Bartlett (1932, page 277):

> When the apparatus came into experimental use, it suffered various modifications of its functional parts which nobody ever thought out very clearly, if at all....Each has developed within its own special social *milieu*, so that a well-instructed onlooker, asked to furnish a rationale for differences in the type of instrument in common use, will often find himself speaking in social, group terms. Yet it is fairly certain that nobody ever put this sort of characteristic before himself as an ideal when he was thinking about the instrument. It simply worked out so in practice.

Understanding how designs "simply work out in practice," how to deliberately manage this process, and what tools facilitate it are themselves basic research questions, but in the domain of socio-technical systems design. The two commitments—which from our computer science laboratory perspective we called 'basic" and "applied" research—are both essential and involve fundamental research. The first primarily occurs in laboratories, like Stanford's Knowledge Systems Lab, where Guidon-Manage was developed. The second must occur in the context in which the system will be used. Conceivably, a given researcher could have a foot in each camp. The underlying claim is that complex, domain-specific computer systems (unlike word processors) can't simply be *delivered* by one community of practice (computer science researchers) to another (medical students and faculty).

Certainly ITS researchers have long been aware of the importance of "people-oriented and organizational issues" (Johnson, 1988). But doing this means much more than "treating subject matter experts as active team members," watching people use our programs in the manner of human factors research, or conducting surveys (Bannon, 1991). To the largest extent practical, we must involve students, teachers, administrators, future employers, and the community as participants in design. We must observe the lives of people using the program, not just their keystrokes. This requires a major leap from the experimental paradigm of testing ready-made programs on a few subjects in a computer laboratory.

Viewed the other way around, *researchers must participate in the community* they wish to influence (Blomberg, et al., in preparation; Jordan, in preparation). As ITS matures, some members of our research community must necessarily broaden their goals from developing representational tools to *changing practice*—changing how people interact and changing their lives. Otherwise, the influence of our tools on everyday life will be delayed and diminished. Recent projects by Lesgold (1992) in apprenticeship training and Anderson (1992) in the Pittsburgh school system illustrate how researchers can become involved.

To see how our choice of commitments influences research, consider Sophie-Game (Brown, et al., 1976). This project involved two teams of students setting faults and debugging electronic circuits. In many ways Sophie-Game fits the collaborative, minimal design that interests researchers today (Roschelle, 1992). Why didn't this design strongly influence ITS research? I conjecture that the design appeared to avoid the problems of central interest to the AI community in the 1970s—automating a tutorial dialogue, representing reasoning strategy, and modeling a student's knowledge. The team approach also complicated modeling what was happening in an individual student's head, an emerging concern at the time[2]. The effect is that our technological goals—exploring the space of what computers could do for instruction—dominated over our education goals. Understanding these different commitments is crucial for researchers who want to develop instructional programs that will change education.

This paper invites ITS researchers to consider shifting from tool design space to everyday life. I begin with a brief introduction to past efforts to embed technology design in social systems. I show how situated cognition theory provides psychological support for the socio-technical systems approach, and secondarily provides guidance about what systems to build. I then elaborate on a design process that involves ITS technology in changing medical education.

2 A Socio-Technical Design Approach

[2]Yet Brown et al. emphasize that the conversations of the students "give us new access to the thinking of the students." Reflecting their tool design interest, they add that this will enable them "to develop more refined automated procedures for diagnosing and correcting student errors" (p. 97). Contrast this computer-centric view with developing aids to help the students understand each other. Incidentally, this is Brown's "ICAI Report Number 1."

Bartlett's remarks about the emergence of design rationales in practice shows that the idea of relating technical design to social systems is not new. In the 1940s especially, this effort became known as "socio-technical design." The difference today is the involvement of anthropologists and computer scientists in making observations and devising tools to manage organizational change (Smith, 1992). We use the same name, but the conception has changed from the perspective of management and systems analysis to include qualitative modeling, restructuring work, and of course information systems.

In applying a socio-technical systems approach to ITS design, we conceive of the unit being designed as comprising physical, social, and information-processing environments (Ehn, 1988; Zuboff, 1988). That is, we are not just delivering a computer box to be placed on people's desks. Additional work is required: We are also designing the room layout and the social organization that will use the technology. This analysis applies equally to classroom and workplace design. Furthermore, we place the group using the computer systems in its own context—the surrounding groups that interact with it, the goals and resource constraints imposed by outside influences. Obviously, this is a much larger problem, which computer scientists or educators alone could hardly claim to handle. Designing computer systems in the context of use requires researchers with multiple perspectives from different disciplines, such as anthropology, linguistics, graphics design, education, organizational management. A central research problem is how to manage and facilitate this collaboration, which must include the computer users themselves.

Ehn (1988) provides a historical description of the development of the socio-technical approach to design. Instead of focusing only on the materials and processes of production (e.g., a manufacturing process), a socio-technical approach studies "both *the technical system* and *the social system* and their *interrelations on the work group level*" (p. 261). In the style of systems analysis, researchers measure variances from the overall system's desired output, attempting to track back the causes of variance to organizational and technical interactions (remaining sensitive to mutual dependencies). Principles for design focus on the level of the "semi-autonomous group rather than individuals." Focus on the group takes into account the individual's sense of challenge, learning, decision-making, recognition, and career development (Ehn, 1988, p. 262). Individual attitudes and beliefs are analyzed from the perspective of membership in the group (Bartlett, 1932; Jordan, 1990; Linde, 1991).

According to Ehn, principles of democracy guide socio-technical system design. The group itself has strong control over its goals, performance assessment, task distribution, membership, leadership, and work procedures. Today we call this *participatory design* (Greenbaum and Kyng, 1991). Overall analysis pays particular attention to coordination *between work groups* and how they internally manage this coordination.

Computer models can provide a basis for reflecting on what is currently happening, measuring cost-benefit of alternative designs, and deliberately restructuring work. On the other hand, computer models can also be used to mechanize human interactions, inhibiting change(Zuboff, 1988). If workers and managers are to remain in control of their work environment, as suggested by the socio-technical approach, they must participate in the design of the computer systems they will use. As Ehn (1988) relates, the emphasis goes beyond the *technology delivery model* of involving managers and "designing systems to fit people." Rather, how do we make it possible for people to participate in the design of their own systems?[3] From the start, we must recognize that both the design process and computer systems we build must consider the inherent tensions and conflicts in social systems; it is easy to adopt an idealistic view of participation, which is blind to the everyday problems of collaboration (Kling, 1991; Hughes, et al. , 1991).

The larger question becomes how technology can increase the possibilities of participation in society: Using computers potentially enhances not just the work of the moment, but a worker's long-term possibilities for participation as a citizen. As a simple example, in the USA we routinely teach high school students about the planets and nebulae, but never mention the principal engineering disciplines and how they relate to the front page of the daily newspaper. Thus, the design process is placed in the context of overall goals for learning (to become a member of multiple communities) and individual growth (Wenger, 1990; Lave and Wenger, 1991).

3 Situated Cognition Motivates and Guides a Socio-Technical Approach

[3]Ehn carries the rhetoric a bit further by suggesting that people "design their own systems." This phrasing emphasizes a democratic approach—for the people and by the people. But it leaves out the outside professionals who can supply design possibilities and facilitate the design process—not to mention the programmers who must do the work! Nevertheless, Ehn is right that the workers must help decide whether computers will be used at all and for what ends.

In itself, the socio-technical approach, with its origins in social and organizational research, has strong implications for how we develop instructional programs. However, this analysis is global, framing the design process, rather than providing guidance about what we should build. A complementary analysis focuses on instructional content and styles of student interaction, relating instructional design to situated cognition theories of knowledge. My interest here is not to reargue the situated cognition case, but to show how it provides a psychological justification and guidance for adopting the socio-technical approach.

My approach to situated cognition is summarized by the aphorism, "Practice cannot be reduced to theory." We will never complete the task of modeling knowledge and human activity (practice) as descriptions of beliefs, reasoning procedures, etc. (theories). Models are of course useful for understanding, teaching, and designing complex systems. But there is no inherent formal structure in practice; what people know and are capable of doing can't be exhaustively described. Psychologically, this means that human memory is not a place where representations are stored (Rosenfield, 1988); that at a base level perception and action are always coordinated without deliberation (Dewey, 1896); and that symbolic reasoning (language) is grounded in pre-linguistic conceptualization (Lakoff, 1987; Edelman, 1992);

The stored-schema model of memory suggests that people can converse and collaborate because they have similar rules for behavior stored in their brains. According to this idea, we speak a common language because we speak by applying grammar rules—each of us carries around copies of culturally common rules in our memories. Similarly, we can collaborate at work because we have copies of schemas (frames, templates, scripts) that describe objects and events. According to this stored-schema model, whenever we perceive and act, we consult these stored descriptions (in a subconscious way), matching and retrieving the appropriate representations. This storehouse view of knowledge suggests that teaching is a process of transferring representations—conveying to a student the necessary facts, models, and procedures that should govern behavior.

Situated cognition research claims that "knowledge is not independent but, rather, fundamentally 'situated,' being in part a product of the activity, context, and culture in which it is developed" (Brown, et al., 1988, p. i). That is, knowledge does not consist of structures that exist independently in the mind (like tools in a shed). Knowledge, as a capacity to behave, *develops during the course of interacting*; this is why we say it is *situated.* Every interaction biases the capacity to coordinate behavior

in an analogous way in the future. Through practice, coordination becomes more direct, more automatic, that is, with less need to create or consult representations of the world or what you plan to do.

Representations are not to be confused with knowledge. Representations are created in the course of activity, during our speaking and writing and imagining. They must be interpreted (through further representing activity) in order to affect behavior. (See (Clancey, 1991a; 1991b; 1991c; 1991d; 1991e; in preparation a; in preparation b; in preparation c; Roschelle and Clancey, in preparation) for supporting examples.)

Grammars describe patterns in human interactions, including especially patterns in the representations we create. But at the core level, we behave without consulting such theories. We don't need recipes for action because:

- It would be recursively impossible (Wittgenstein's argument of needing rules for interpreting what the rules mean (Tyler, 1978)),
- It would confine our behaviors to grammars that control every action (Cohen's argument that AARON can't draw original pictures if it must follow grammars supplied by a human designer (Clancey, 1991a)),
- The brain doesn't work by interpreting stored facts and programs; every behavior is a novel coordination of perceptual and motor systems (Edelman, 1992; Clancey, 1991b; Clancey, in press).

The key idea is that *knowledge is a capacity to interact.* Interaction includes how I move and talk with people, how I manipulate materials, and how I engage in private acts of representing to myself (e.g., imagining and planning my daily activities). Processes of interaction come into being during interaction itself—as opposed to being predescribed in stored schemas and code that are merely retrieved and executed as a program. Interaction can be fruitfully described at different levels, including social, psychological, and neural. At each level, emergent structures are constraining development, but adaptation (learning) is occurring with each construction, each pen stroke, each sentence, and each social maneuver. That is, learning is a primary phenomenon, on which secondary, reflective reasoning is based. Social scientists emphasize that knowledge resides in the interaction because it has no existence apart from its realization in activity (i.e., knowledge is not a stored thing). We can of course describe what people do and reflect on our intentions and patterns of behavior, but such representations are always the product of our interactions (even as we speak or type), not the inner mechanism that drives our behavior.

The punch line is that social systems continuously develop. This is true especially because people are individually changing as they become more efficient through practice. Secondarily, people reflect on what they are doing, and add insights that lead to physical, social, and technological restructuring of the system. At another level, through the interactions of its members, a work group adapts to its changing economic and political environment.

This means that in automating work or formalizing a subject domain there is no fixed target to build into the computer system. We can describe the community's models, work procedures, and conversational patterns. But these theoretical descriptions are not the basis of the next actions people will take. Furthermore, even when they are articulated by the members of the community, the meanings of these representations—and hence their implications for orienting behavior—are constantly prone to change: “The use of words is itself a creative act, somewhat physical, that produces meanings that did *not* exist in prior thought” (Goodman, 1971).

As an example, consider this electronic mail exchange between two members of a conference program committee:

> *Member of committee to other reviewers:*
> If anyone cares, what the chairman's policy translates into is the following:
> If any paper has width x length x pages > 567 square inches, reject it.
>
> *Another member responds:*
> This seems a bit mechanical to me; for one thing maybe the author included too many diagrams, or didn't shrink them as much as possible or any of a dozen other things.
>
> This was discussed at great length at last year's program committee through the mail (several dozen messages). It was also discussed by the conference committee....I don't think that more discussion will help; people should simply try to be reasonable.

The experienced member of the committee suggests that they are not expected to follow a rule mechanically. There is no formal specification that defines once and for all whether a paper should be rejected because of length. Reviewers are told to “be reasonable.”

Social scientists claim that such examples provide evidence for the social construction of meaning, the open nature of concepts. People don't operate and aren't

expected to operate mechanically (i.e., simulating a computer). When people interpret rules, they must do so in an *ad hoc* manner—ultimately there are no further definitions to rely on, you must simply act. (For example, should a reviewer measure quotations with indented margins separately? There is no end to special cases because people didn't *create* their papers by following rules mechanically!) The ultimate advice must always be, “Do something, be yourself, be reasonable.” This is necessary because you have no stored rules or schemas to consult. And the representations you do generate (e.g., your own paper-reviewing rules of thumb) must ultimately be acted upon without consulting further representations.

In particular, the use of representations on a computer screen must be interpreted by people in order to be used. This sometimes creates new tasks of communicating with more proficient workers and inventing work-arounds (Zuboff, 1988; Wenger, 1990). This interpretation process is one of conceptualization, involving to some degree redefining what constitutes the goals, tasks, and values of the group (Wynn, 1991). Every action, including every act of generating and interpreting representations, is adapted and novel; it is the product of interacting. Consequently, ongoing work is a process of learning, refining, and adjusting. To automate this human constructive activity by template retrieval and matching is to lose opportunities for change; to convey schema-models as *equivalent* to what people do wrongly biases a student’s education (Schön, 1987).

Neither procedures nor their interpretation can be imposed without inhibiting innovation. Practice (what people say and do in the course of their interactions) cannot be reduced to theory (grammars, rules, and scripts orchestrating behavior):

—Every coordination of seeing and doing in people is a new neural construction, not dictated by a stored description of how it will appear, is defined, or is constrained by other objects and properties (Edelman, 1987; Freeman, 1991).

—Concepts are not things, but ways of seeing and talking that recur (Bamberger and Schön, 1983).

—The similarities of the world and behavior are in our perception, not objective (Tyler, 1978; Lakoff, 1987; Gregory, 1988; Winograd and Flores, 1986).

—Recurrence is possible because processes become coupled. Routines are not rote occurrences, but analogically similar, adapted from similar coordinations and perceptually elided by the observer (Bartlett, 1932).

In simple terms, the world and what people do are inherently messy. Yes, there are patterns because interactions recur: People become more coupled to their environments (you reach for things without having to represent to yourself where they are). But to say that a person's behavior is fully equivalent to what a schema-based computer program can do—that practice can be reduced to theoretical description—is to leave out the work that people must do when they use representations, when they interpret each other's words, when they decide moment-by-moment what constitutes information and what values frame their community (Winograd and Flores, 1986). In the case of the paper reviewers, every judgment is part of the process of constructing what the community believes and how it operates. Because the materials and goals upon which the reviewers operate cannot be exhaustively predefined—there is no one right, complete theory—the reviewer's procedures and perception of the task are always adapting. The use of computers in the workplace and schools must leave open these possibilities for change.

4 How We'd Develop Guidon-Manage Today

Situated cognition helps us better relate theoretical models like Neomycin's disease taxonomy to medical practice, so that learning and using such models occurs within the context of a community of practice (as opposed to being handed over as objective facts, existing independently of human modelers and practitioners). Looking back at our attitudes and methods in developing Guidon-Manage, I see many changes in my perspective:

- Participating with users in multi-disciplinary design teams, versus viewing teachers and students as my subjects;
- Adopting a global view of the context in which a computer system will be used, versus delivering a program in a computer box;
- Being committed to provide cost-effective solutions for real problems, versus imposing my research agenda on another community.
- Facilitating conversations between people, versus only automating human roles;
- Realizing that transparency and ease of use is a relation between an artifact and a community of practice, versus an objective property of data structures or graphic designs;

— Relating schema models and ITS computer systems to the everyday practice by which they are given meaning and modified, versus viewing models and programs as constituting the essence of expert knowledge, to be transferred to a student;

— Viewing the group as a psychological unit, versus modeling only individual behavior.

I elaborate on these perspectives below.

4.1 Participatory design

A physician-teacher and a medical student were always part of the Guidon-Manage team. But they were called into our offices as consultants on our design project. We spoke to a medical school professor, to see how he used computer programs in his classes. But we thought of our work as futuristic, not for immediate use. Indeed, part of the glamour and allure of being an AI researcher was that we were designing systems that would change people's lives in the future. As I have already described, the model of basic research in the Knowledge Systems Lab of the 1970s and 80s was that we would develop techniques and publish. We were not concerned with practical issues (e.g., how users could afford the computers). Applied research would be undertaken by industry and teachers, who would read about our ideas and reduce them to practice.

ITS researchers aiming to develop programs that people use today must reject the serial, delivery view of software technology. Developing computer systems over the short term (6 months to a year), that are affordable, and that people use enthusiastically as part of their everyday endeavor, becomes a basic research problem. It is fine to develop a tool kit, to explore the theory of computation, and to invent new programming languages. But if I am committed to developing instructional programs for students to use, I must work with teachers and *design for today*, not the next decade.

The socio-technical systems approach suggests that users (students and teachers) must participate from the very beginning. But also, because the world is a messy place (we cannot specify once and for all how the world or people work), we must develop our designs in the course of use, incrementally, with relatively quick periods of use, observation, reflection, and redesign. That is, our computer systems, as artifacts that fit into people's lives, must develop in a context that includes the user's

everyday adaptations[4]. The context of use is a moving target, changed especially by technology itself. Users can imagine and to some degree anticipate how our designs will be used and the implications for other aspects of the social, physical, and information processing system (Greenbaum and Kyng, 1991). But in practice, all the team can do is make a best guess, reconfigure the overall system, and observe what happens.

Developing Guidon-Manage today, we would aim to develop a community of teachers and students who shared our commitment to exploring the use of computers and knowledge representations in teaching medical diagnosis. Together, we would design the system around the curriculum, interactions with between people in different physical environments, and daily routines. We would essentially be *designing interactions* within the context of the medical school's social, physical, and information technology environment, not designing a computer system conceived in isolation sitting on computer scientists' desks.

This may all appear obvious, but the change in mind set required is radical. For example, in their otherwise impressive study of participatory design for ITS, Murray and Woolf (in press) referred in a draft report to "the three participants and the knowledge engineer." Until the knowledge engineer is viewed by everyone as a participant, we are not treating the students and teachers as colleagues, but as subjects—subject to our designs.

4.2 Global view of context of use

In developing Neomycin, we visited classrooms and followed teachers in the medical clinic. Our perspective was to develop the best model, the *correct* model, of diagnostic reasoning. We were aware that physician behavior was influenced by the context (e.g., being in an emergency ward versus having time to think while reflecting in a computer scientist's office). But we viewed differences in behavior as variations caused by conditional variables of time pressure, uncertainty and availability of data. We believed there to be one underlying procedure stored in the physician's memory, which, with its conditional actions, was applied in every setting. In effect, we believed the mind to have a stored program for doing diagnosis that existed independently of where the program was applied. Our goal was to represent this diagnostic procedure and medical facts. The places where physicians and students

[4]It is tempting to object here that computer programs simply aren't the kind of things that are adapted by people who use them. Programs must simply be accepted and used as they are given. This is precisely the attitude and method a socio-technical approach calls into question.

worked were just places where we could find them *applying* their knowledge. Notice how such cognitive models are biased by computer programming concepts—knowledge is presumed to be something static that is applied when we reason, just as we APPLY a lisp function; no learning is required.

Our study of physicians and medical students did not consider their lives in any general sense. Although we thought we were developing an instructional program for students to use, we gave almost no consideration to where or when the program would be used. At one point, we visited the Stanford Medical School Library, to consider the logistics of placing a Xerox D-Machine there. Our idea was that students might use the program after learning related material in classes, or they might try to "run a patient" through the program that they had just examined in the wards. We assumed that using Guidon-Manage would have priority in the students' lives. The rest were details to be worked out once the program was ready for use.

Of course, we were not funded to provide a computer system for routine student use, never proposed to do so, and never thought it was an activity that computer science researchers would directly engage in. We were indeed well aware of the controversy in the Knowledge Systems Laboratory when Shortliffe committed to developing a program for routine use in the medical clinic (Oncocin), which appeared to be merely an application of AI research. Here again our stored-schema theory of knowledge is manifested: Basic research creates knowledge, applied research is merely its transfer to another setting. Just as applying Neomycin's knowledge in practice was to be merely the application of static, stored knowledge, moving Guidon-Manage to the medical school wouldn't require new theories of medical subject matter or instructional strategy. Paralleling the distinction we drew between schema-theories and the pragmatic messiness of what people actually do, we distinguished between our effort to create computer modeling methods and the messy political and economic problems of developing systems that students would actually use. Thus, the idealized, grammatically-constrained mechanism of AI programs reflects our alienation from everyday interaction and *the development of ideas in practice*.

As an example of the applied research required to relate ITS design to the global context, we need to understand when and where learning already occurs in the medical school (in the student lounge? in lunch conversations? in evening study sessions? in the library?). Medical students are pressed for time, but how do they actually use their time? We simply assumed over the years that we were creating a

system that would save time, so all students would want to use it. But do students need more time for physical recreation? To see real patients? To study physiology? To meet with friends to discuss what is happening in class? In retrospect, we cannot say that we were truly committed to helping students because we knew next to nothing about the student's lives, their priorities, and their choices. Shifting perspective to the context of use means *working with students and teachers in their setting*—not just calling them into the computer science lab to work *with us*. Here the ethnographic approach of participant observation, living with the other culture, is central (Jordan, in preparation).

4.3 Commitment to cost-effective solution of real problems

Developing Guidon-Manage in the 1980s, we were most concerned with publishable AI results. We felt comfortable working in the medical domain because of its relevance to society. We justified the diagnostic focus with our sponsors, the Office of Naval Research, in terms of its applicability to maintenance of electro-mechanical systems[5]. As measured by our influence on expert system and ITS architectures, Neomycin and Guidon-Manage research certainly succeeded from ONR's perspective.

But for all our good intentions, the fact remains that we were not devoting our energy to helping medical students and teachers (or patients). Our activity was always peripheral to the medical school community, though you might think from our enthusiasm that we were turning the place upside down. Now it is clear that to develop complex systems in practice, I must back off considerably and greatly broaden my concern. I must not go into the medical school as just an AI researcher or even a computer scientist. I must enter as a citizen, a manager, someone concerned about the overall process of medical care and education. I must adopt a global view of the setting I hope to influence. I must learn and understand what is happening in that world, and then apply my skills accordingly.

As an example of the change in mind set that is required, I would have every member of the Guidon-Manage team today carry out a thought experiment to examine their motivations. I would ask, "Should I give you $1 million to spend in the Stanford Medical School, where could that money have the greatest effect? What are the priorities of the patients, the nurses, the cafeteria staff?" I want the members of the team to know where an instructional computer program fits in that world. How do

[5]We were also funded by the Macy Foundation in a medical cognitive science program.

our efforts rank in terms of what these people care about and (perhaps different) how that money could be best spent? In effect, I want computer scientists to know where they stand in the community. Would paying higher salaries to nurses have a greater effect on patient care? Do students simply need word processors? Should we subsidize apartments for the interns adjacent to the hospital? As citizens, we must have integrity in relating our technology to the specific community we are trying to change.

I don't argue that needs and possibilities can be so easily ranked, or that self-promotion and salesmanship isn't necessary. But I want computer system developers (and all scientists) to have informed opinions about where their contributions stand. We must be ready to argue how our contributions fit into the total picture. This is of practical concern, for we must compete for research money and society's attention. We may decide that we have nothing to contribute as change agents, and choose instead to focus on tool design (and then perhaps turn down our technology hype). But for selfish reasons as well, we must understand the non-technical reasons why our efforts may fail. Enlisting appropriate participation by social scientists, management, and other members of the community ("buy-in") may make all the difference in developing a successful system (Leonard-Barton, 1987; Mumford & MacDonald, 1989).

4.4 Facilitate vs. automate conversations

Computer scientists generally view computer programs in terms of automation or simulation. The idea is to replicate in the computer processes that naturally occur in the physical or human world. This view is biased of course by the traditional use of machines to automate processes that people otherwise would do manually. The very idea of a machine is of something that does some task in a regulated, autonomous manner. In AI research this led to machines that solve problems, interpret information, and speak.

Another perspective is that of the computer as a tool, more like a blackboard or a drawing tablet. Rather than an agent that speaks in the role of a person, one conceives of computer systems that provide a medium for people to express themselves. Paint programs and CAD systems are familiar examples. Of special interest, not often emphasized in the past, is a computer tool that helps people carry on a conversation with each other (Roschelle, 1992; Roschelle and Clancey, in press). For example, a simulation model can enable one person to show the other what will happen under certain conditions. People can point on the screen and discuss the use and meaning of

representations. Expert systems might also be used in this way—consider using R1 to facilitate a conversation between a sales person an a client rather than to replace part of the sales person's job.

Part of our fixation on "one-computer per student" may have developed from the view that the problem in the schools was inadequate individualized instruction. But recalling Sophie-Game (Brown, et al., 1976), having the students work alone may not be optimal either. The socio-technical approach reminds us that people will always be conversing in the classroom and workplace, and often we learn from each other: Could we use Guidon-Manage as a centerpiece to facilitate such conversations? Our perspective on tool design, shaped by primary commitments to automating conversations, replacing a human participant, never led us to consider such questions.

4.5 Non-objective view of representations

One of the key ideas of ITS research in the 1970s was the importance of "glass box design." We believed that explanation programs could make the operations of a computer program transparent. In contrast with a black box, with hidden mechanisms, we could reveal the workings of an expert system. This idea was further elaborated in Neomycin by representing the diagnostic procedure declaratively, separate from the models of diseases ("domain knowledge")[6].

Situated cognition research (Section 3) reminds us that transparency is in the eye of the beholder. Without an appropriate medical background, and understanding the context in which Neomycin would be used, Neomycin's explanations are not comprehensible. Wenger has developed this idea further to point out that transparency can be usefully viewed as a relation between an artifact and a culture (Wenger, 1990). Transparency isn't an objective, inherent property of an artifact, but *a relation* between an artifact and the knowledge and interactions of a community of practice: You might not understand Neomycin's statements, but your co-worker may be ready and able to explain. For a group of people working together, a design may be comprehensible that would not be transparent to individuals alone (recall Bartlett's remark that design rationale is socially constructed). Evaluating Guidon-Manage therefore entails viewing it within the socio-technical context of the medical school, not just one student sitting in front of the machine, puzzling out the program in isolation.

---

[6]"Declaratively" means that the structures of the program are formatted and annotated so they can be interpreted by multiple processes (e.g., by a compiler, an explanation program, a student modeling program) (Clancey, 1992).

The idea of cultural transparency calls our attention to the creative aspect of sense-making, in which a community is continually theorizing about its goals, values, and the meaning of its representations. In particular, we didn't sufficiently consider in designing Guidon-Manage (or Mycin) the theorizing that occurs everyday in the medical school. Theoretical representations belong to and are modified by teachers and students, at all levels of capability, not just physician specialists. Our concept of explanation was again of transferring knowledge, rather than a process of mutual learning and negotiation (Rommetveit, 1987; Clancey, in preparation d). An alternative view of knowledge engineering, in which representations serve as a medium for communication within a community of practice, rather than being delivered from outside, is developed by Stefik and Conway (1988). This suggests that we do something other than merely ITS authoring tools to the medical faculty or providing a fixed knowledge base to be taught. It also suggests that we develop tools for a group of teachers working together.

Research in the context of use is required to determine to what extent our present technology allows involving students and teachers in the computer modeling process. We need to better understand what is happening today: By what interactions and in what media are representations of medical knowledge currently created and used in the medical school? Rather than turning over prescriptive models of what people in another community should believe and do (exemplified by the title of my dissertation, "transfer of rule-based expertise"), we reframe the ITS design process in terms of creating tools for stating and testing models. Again, our view that knowledge was objective, pre-conceived truth—transferred from expert to knowledge engineer to student to application setting—obscured how learning to be a physician is related to learning to be a scientist. To explore this further, we consider how models are related to everyday practice.

4.6 Relating models to everyday practice

Apart from courtesies of the bedside manner, it was difficult to conceive what a medical student should be taught, other than Mycin's rules. We didn't realize that the complement of Neomycin—how the model relates to the unformalized world of medical practice—is an essential part of what a student needs to know. What explanations could help students and consultation users become more aware of the quality of their work, know when to question what they are doing, and generate ideas for changing what they are doing? In other words, how can knowledge representations help students to be *reflective practitioners* (Schön, 1987)? One

approach, following Dewey, is to make the curriculum (a knowledge base) an object of inquiry.

As an example, consider Neomycin's disease taxonomy. Today we view such representations not as a product to be delivered to a student, but as a partial model of a practice. Besides learning the various diseases and their relations, we would want the student to consider the following:

- ❏ Why do we taxonomize diseases?
- ❏ What gets glossed? How do we know when this stereotypic view of physiological processes is misleading?
- ❏ Who knows this taxonomy; what is its origin?
- ❏ What is the nature of disagreements; how do they arise; how are they settled?
- ❏ How are taxonomies related to medical research? Are they becoming unnecessary as we develop better mechanistic models?
- ❏ What are good ways to keep a taxonomic perspective up-to-date?

By this view, knowledge is knowing how to live in a community, not just static facts or procedures that represent what people do. This is what we want the student to learn and what our computer tools could support (Lave and Wenger, 1991; Schön, 1987; Clancey, in preparation c; Greenbaum and Kyng, 1991).

4.7 The group as the psychological unit

Frederic C. Bartlett pioneered studies relating individual and group behavior. His memory experiments in particular suggest that cognition is, in his terms, a "socially constructive" process (Bartlett, 1932, pps. 274-280):

- ❒ Coordination functions in activity, not in the individual mind (again, practice is grounded in what people do, which is never fully deliberated and has emergent effects outside of individual control);
- ❒ Contributions that stand must be part of a group trend (no individual plan or goal can strictly control a community, without mechanizing behavior);
- ❒ An individual acquires greater influence in a complex community (because there are more possibilities for variation and ways of causing change);
- ❒ Swift insight changes the group, but details in working out ideas emerge, dependent on the "form and trend of the group before the achievement is effected" (learning is a property of the community as a whole);

- ❒ Design rationale for artifacts emerges from practice (a design rationale is not always a preconceived plan used to create an artifact);
- ❒ Modifications to an instrument develop in practice (and so cannot be attributed exclusively to an individual or a linear aggregation of individual contributions).

To explicate these points, Bartlett draws a strong parallel between social development and an individual's design activity. First, an artist isn't merely executing a preconception, but necessarily improvises, reperceiving the ongoing trend of his drawing, interpreting its force and meaning, and incrementally adding or reshaping what is there. "Having started his design, the rest of the figure must fall into a certain harmony of outline and balance of parts which, of course, limit individual choice." That is, the artist's own drawing action is constrained by the trends he has himself produced. Not just any contribution will do. Furthermore, the characteristics of the drawing are themselves a realization of cultural practices, values, and activities. Thus, to understand the origin and influences of individual contributions, we must view them as interactions within a dynamically changing social environment.

This analysis suggests that we complement models of individual knowledge and behavior by *models of a community's knowledge and behavior*. By this perspective, learning for the individual is *becoming a member of a community of practice* (Lave and Wenger, 1991). This should be contrasted with the dominant view in ITS research, epitomized by our student modeling programs, in which descriptions of learning are usually confined to a single student's interaction with a computer program.

In terms of Guidon-Manage, we would be interested in how the community of students is changing in the course of an academic year and throughout medical school. How are attitudes and beliefs constructed by informal networks, friendships, relations between students and professors (Eckert, 1989)? How does this group, through its various activities, develop a shared set of values, goals, and collaborative roles, relative to their interactions with surrounding university, hospital, and patient communities?

These considerations may at first appear to be far afield from learning a disease taxonomy. The essence of the socio-technical perspective is to view learning in terms of the processes of interactions, the demands, and the resources of the larger system that includes existing technology, language, interpersonal interactions, and physical environment. To give a simple example, we could not pretend to present Guidon-

Manage as a solution to the "medical instruction problem" without taking into account the trends of the community: Where does ITS technology fit within their emerging concerns? What dialogues about change are currently occurring in the medical school (e.g., about overhead costs, about the cost of medical care, about national health insurance)? How will interactions between people in their everyday activities positively or negatively influence their use of Guidon-Manage? Who views his or her job as including curriculum design or managing computer use in the medical center? Without these considerations, we cannot complain if our programs are not used or people fight our very involvement in their lives.

5 Conclusions

I have spent most of the past four years reconsidering the assumptions that directed my AI research. I have concluded that the exclusively individualistic view of cognition as something that occurs inside individual brains is useful for instruction, but a narrow conception of knowledge. Observing the work of social scientists studying the workplace (e.g., Jordan, in preparation) and designing computer systems (e.g., Ehn, Greenbaum and Kyng), I have concluded that as a computer scientist interested in applications programming, I must turn my work upside-down. I must start with the user environment, not computer science ideas. Rather than developing systems inside a computer lab and delivering to users, I must develop within the context of use. The idea that I could *demonstrate* a medical instructional program to teachers in a computer science office now seems ludicrous to me. I view system development as occurring in the larger system that includes people with different perspectives: managers, graphic designers, workers, students, anthropologists, programmers. Research questions shift to *how to coordinate this dialogue*, particularly by using prototype and modeling tools. In effect, these tools and abstractions enable fundamental shifts in responsibility and authority from computer scientists to other members of a design team, changing the design process. Again, the shift is from the primary focus on automating "acquisition of expert and teacher knowledge" to tools for facilitating a dialogue between designers of different disciplines.

If I really care about developing programs that people will want to use, I must shift to a short-term, incremental approach. Complex systems must develop within the context in which they will be used. Ideally, this means developing the simplest possible systems, which are of value from the very start. What are the most basic

computer tools that could serve the community I am trying to help? I must seek to build my systems around core capabilities that can be developed early on.

From this perspective, I work with anthropologists and other social scientists because I want the broadest possible understanding of how my work will fit within the larger socio-technical system. If I am committed to helping people in some community, I need to keep relating my designs to their lives. I want my efforts to be honest; I am not just advocating and selling computers and software. I want my work as a computer scientist to have integrity. Besides belonging to the computer science research world, I am a citizen in a larger community that includes anthropologists, patients, and educators.

With this larger connection, my research interest necessarily shifts from developing tools, such as modeling languages, in which I explored the space of what computers can do. Obviously, we still want people to keep doing that. But developing complex new mechanisms is not my focus today. As a member of the design team, I remain a promoter of computer science ideas, but I am tuned to what designs will make a difference, rather than what will appear in my next AI publication. I ask, "What combination of existing methods can be combined and extended in a cost-effective way to bring value in the next six months or year?" That's my new puzzle. My design is more constrained, my research is *inherently empirical*. Consequently, I must step back, observe and listen in a new way, and find new opportunities to contribute. Correspondingly, my publication audience shifts to the *Journal of Medical Education* or perhaps the *Anthropology of Work Review*. Indeed, through publications, conferences, and research projects, I will participate in multiple communities of practice. The key realization is that developing useful software involves participating in the user's community—a shift from the laboratory research perspective.

To summarize, a socio-technical approach to software design includes, but goes beyond, traditional views:

— Transfer view of modeling
— Program-in-a-box view of designed system
— Delivery view of development
— "Subjects" view of user community
— Input/Output view of evaluation
— Automation view of computers
— Objective view of transparency

We need a better understanding of the relation between representations and knowledge, the role of the social and physical environment on learning, how uses of computer artifacts develop in practice, and how models are created and adapted in unusual situations. How computer scientists can work with users in participatory design to bring about these shifts is a research problem, which must develop in practice. The experience of Ehn (1988), Kukla, et al. (1990), and Greenbaum and Kyng (1991) provides heuristics for engaging users in design questions, such as using familiar design tools, using story-boards to present models of knowledge and activity, and helping workers envision future work situations (e.g., to anticipate conflicts and benefits). From the perspective of ITS, we go beyond knowledge acquisition authoring tools to consider the roles and interactions of computer designers and teachers. In the lingo of Greenbaum and Kyng, we view design as *cooperative action*, an ongoing interaction. How we set up the design process will determine what we build.

To recap, ITS research began in the 1970s with a psychological theory: Knowledge consists of representations; learning occurs when applying representations to problems. But theories of situated learning suggest that knowledge, as a capacity to behave, is always a novel construction that develops with every action. Related theories of memory suggest that these constructions are analogous to previous ways of seeing and moving because they are actually composed of previously active neural maps.

Trying to find a more useful level of description for instructional design, we jump up from the neural to the social plane—we consider human action in relation to the practices of a community. For example, we view motivation in terms of not just internal curiosity, but the emotional aspects of belonging to a community. We do not bypass the knowledge-description level, but place such theories and how they are used within the social context in which people create and interpret representations everyday. We don't view representations as the underlying substrate that stores human beliefs and controls behavior, but as a medium for reifying and coordinating interactions in the social and physical world.

We shift to viewing computer system design as a group activity involving different communities of practice (e.g., users and programmers). Bartlett saw this relation and reminded us that what artifacts mean, how they are used, and how they are rationalized will develop in the course of social interaction. With this larger view of

how representations and artifacts (and hence knowledge) develop, we move ITS to the socio-technical arena.

Acknowledgments

This paper is the result of years of interactions with my colleagues at IRL: Penny Eckert, Shelley Goldman, Jim Greeno, Rogers Hall, Gitti Jordan, Ted Kahn, Charlotte Linde, Jeremy Roschelle, Susan Stucky, and Etienne Wenger. Detailed comments by the reviewers, Warren Sacks and Dennis Newman, as well as attendees at the Montreal ITS-92 conference were especially useful for articulating the distinction between the two research commitments. I am grateful for the support and encouragement of my AI colleagues, John McDermott of DEC and Rolf Pfeiffer of Zurich. Chuck Kukla's designs at DEC showed me the generality of Winograd's approach to using computers, and how to put it into practice. Funding has been provided in part by grants from the Xerox Foundation and the National Science Foundation. Guidon-Manage research was funded by the Office of Naval Research and the Macy Foundation.

References

Anderson, J. 1992. Intelligent tutoring and high school mathematics. In C. Frasson, G. Gauthier, and G.I. McCalla (eds), *Intelligent Tutoring Systems, Second International Conference, ITS '92,* Berlin: Springer-Verlag, pp. 1-10.

Bamberger, J. and Schön, D.A. 1983. Learning as reflective conversation with materials: Notes from work in progress. Art Education, March.

Bannon, L. 1991. From human factors to human actors, In J. Greenbaum and M. Kyng (eds), *Design at Work: Cooperative design of computer systems.* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 25-44.

Bartlett, F.C. [1932] 1977. *Remembering—A Study in Experimental and Social Psychology.* Cambridge: Cambridge University Press. Reprint.

Blomberg, J., Giacomi, J., Mosher, A., and Swenton-Wall, P. (in preparation) Ethnographic field methods and their relation to design. To appear in *Participatory Design: Perspectives of System Design*, Lawrence Erlbaum.

Brown, J.S., Rubinstein, R., and Burton, R. 1976. *Reactive learning environment for computer-assisted electronics instruction.* ICAI Report No. 1, BBN Report No. 3314.

Brown, J. S., Collins, A., and Duguid, P. 1988. *Situated cognition and the culture of learning.* IRL Report No. 88-0008. Shorter version appears in *Educational Researcher*, **18**(1), February, 1989.

Buchanan, B. G., and Shortliffe, E. H. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Heuristic Programming Project.* Reading: Addison Wesley,

Clancey, W. J. 1988a. Acquiring, representing, and evaluating a competence model of diagnosis. In M. Chi, R. Glaser, & M. Farr (eds), *The Nature of Expertise*, pp. 343-418.

Clancey, W. J. 1987. *Knowledge-Based Tutoring: The GUIDON Program.* Cambridge: MIT Press.

Clancey, W. J. 1988b. The knowledge engineer as student: Metacognitive bases for asking good questions. In H. Mandl, & A. Lesgold (eds), *Learning Issues in Intelligent Tutoring Systems*, Springer-Verlag.

Clancey, W.J. 1991a. Why today's computers don't learn the way people do. In P. Flach and R. Meersman (eds), *Future Directions in Artificial Intelligence.* Amsterdam: Elsevier, pp. 53-62.

Clancey, W.J. 1991b. Review of Rosenfield's "The Invention of Memory," *Artificial Intelligence*, **50**(2):241-284, 1991.

Clancey, W.J. 1991c. The frame of reference problem in the design of intelligent machines. In K. vanLehn (ed), *Architectures for Intelligence: The Twenty-Second Carnegie Symposium on Cognition*, Hillsdale: Lawrence Erlbaum Associates, pp. 357-424.

Clancey, W.J. 1991d. Invited talk. *AI Communications—The European Journal on Artificial Intelligence* **4**(1):4-10.

Clancey, W.J. 1991e. Situated Cognition: Stepping out of Representational Flatland. *AI Communications—The European Journal on Artificial Intelligence* **4**(2/3):109-112.

Clancey, W.J. 1992a. Model construction operators. *Artificial Intelligence*, **53**(1):1-115.

Clancey, W.J. 1992b. Representations of knowing—in defense of cognitive apprenticeship. *Journal of AI in Education*, **3**(2):139-168.

Clancey, W.J. (in preparation a). *Interactive control structures: Evidence for a compositional neural architecture.* Unpublished manuscript.

Clancey, W.J. (in preparation b). A Boy Scout, Toto, and a bird: How situated cognition is different from situated robotics. A position paper prepared for the

NATO Workshop on *Emergence, Situatedness, Subsumption, and Symbol Grounding.* To appear in a special issue of the AI Magazine, Brooks and Steels (eds).

Clancey, W.J. (in preparation c). The knowledge level reconsidered: Modeling socio-technical systems. To appear in *The International Journal of Intelligent Systems*, special issue on knowledge acquisition, edited by Ken Ford.

Clancey, W.J. (in preparation d). Notes on "Epistemology of a rule-based expert system" and "Heuristic classification." To appear in a special issue of most influential papers of *Artificial Intelligence.*

Clancey, W.J. (in press). Situated action: A neuropsychological interpretation. To appear in *Cognitive Science*.

Dewey, J. [1896] 1981. The reflex arc concept in psychology. *Psychological Review*, III:357-70, July. Reprinted in J.J. McDermott (ed), *The Philosophy of John Dewey*, Chicago: University of Chicago Press, pp. 136-148.

Eckert, P. 1989. *Jocks and Burnouts*. New York: Teachers College Press.

Edelman, G.M. 1987. *Neural Darwinism: The Theory of Neuronal Group Selection.* New York: Basic Books.

Ehn, P. 1988. *Work-Oriented Design of Computer Artifacts*. Stockholm: Arbeslivscentrum.

Floyd, C. 1987. Outline of a paradigm shift in software engineering. In Bjerknes, et al., (eds) *Computers and Democracy—A Scandinavian Challenge*, p. 197.

Freeman, W. J. 1991. The Physiology of Perception. *Scientific American*, (February), 78-85.

Goodman, P. 1971. *Speaking and Language: Defence of Poetry.* New York: Vintage Books.

Greenbaum J. and Kyng, M. 1991. *Design at Work: Cooperative design of computer systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Gregory, B. 1988. *Inventing Reality: Physics as Language* . New York: John Wiley & Sons, Inc.

Hasling, D., Clancey, W. J., & Rennels, G. 1983. Strategic explanations in consultation. *International Journal of Man-Machine Studies*, **20**(1):3-19.

Hughes, J. Randall, D., and Shapiro, D. 1991. CSCW: Discipline or Paradigm? A sociological perspective. In L. Bannon, M. Robinson, and K. Schmidt (eds), *Proceedings of the Second European Conference on Computer-Supported Cooperative Work.* Amsterdam, pp. 309-323.

Johnson, W.B. 1988. Developing expert system knowledge bases in technical training environments. In J. Psotka, D. Massey, & S. Mutter (eds), *Intelligent Tutoring Systems: Lessons Learned,* Hillsdale, NJ: Lawrence Erlbaum Publishers, 21-33.

Jordan, B. 1990. Technology and the Social Distribution of Knowledge. In J. Coreil and D. Mull (eds), *Anthropology and Primary Health Care*. Westview Press, Boulder, pp. 98-120.

Jordan, B. (in preparation). New research methods for looking at productivity in knowledge-intensive organizations. To appear in v.D. Parunak (ed), *<title to be determined>*. Industrial Technology Institute Technical Report.

Kling, R. 1991. Cooperation, coordination and control in computer-supported work. *Communications of the ACM*, **34**(12)83-88.

Kukla, C.D., Clemens, E.A., Morse, R.S., and Cash, D. 1990. An approach to designing effective manufacturing systems. To appear in *Technology and the Future of Work*.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* Chicago: University of Chicago Press.

Lave, J. and Wenger, E. 1991. *Situated Learning: Legitimate Peripheral Participation.* Cambridge: Cambridge University Press.

Leonard-Barton, D. 1987. The case for integrated innovation: An expert system at Digital. *Sloan Management Review*, Fall, pp. 7-19.

Lesgold, A.M. 1992. Going from intelligent tutors to tools for learning. Abstract appears in C. Frasson, G. Gauthier, and G.I. McCalla (eds), *Intelligent Tutoring Systems, Second International Conference, ITS '92,* Berlin: Springer-Verlag, p. 39.

Linde, C. 1991. What's next? The social and technological management of meetings. *Pragmatics*, **1**, 297-318.

London, B., and Clancey, W. J. 1982. *Plan recognition strategies in student modeling: Prediction and description.* AAAI-82, pp. 335-338.

Mumford, E. and MacDonald, W. B. 1989. *XSEL's Progress: The Continuing Journey of an Expert System.* New York: JohnWiley & Sons.

Murray, T. and Woolf, B. 1992. Encoding domain and tutoring knowledge via a tutoring construction kit. *Proceedings of AAAI-92.*

Norman, D. A. 1988. *The Psychology of Everyday Things.* New York: Basic Books.

Richer, M., and Clancey, W. J. 1985. GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. *IEEE Computer Graphics and Applications*, **5**(11):51-64.

Rodolitz, N. S., and Clancey, W. J. 1989. GUIDON-MANAGE: teaching the process of medical Diagnosis. In D. Evans, & V. Patel (eds), *Medical Cognitive Science*. Cambridge: Bradford Books, pp. 313-348.

Rommetveit, R. 1987. Meaning, context, an control: Convergent trends and controversial issues in current social-scientific research on human cognition and communication. *Inquiry,* 30:77-79.

Roschelle, J. and Clancey, W. J. (in press) *Learning as Social and Neural.* Presented at AERA91, Chicago. To appear in a special issue of the *Educational Psychologist.*

Roschelle, J. 1992. *Students' Construction of Qualitative Physics Knowledge: Learning about velocity and acceleration in a computer microworld.* Doctoral Dissertation in Education, University of California at Berkeley. IRL Technical Report 92-0003.

Rosenfield, I. 1988. The Invention of Memory: A New View of the Brain. New York: Basic Books.

Schön, D.A. 1987. *Educating the Reflective Practitioner*. San Francisco: Jossey-Bass Publishers.

Smith, G. 1992. The new realism in office systems. *Business Week,* June 15. pp. 128-133.

Stefik, M. and Conway, L. 1988. Towards the principled engineering of knowledge. In R. Engelmore (ed), *Readings From the AI Magazine, Volumes 1-5, 1980-85.* Menlo Park, CA: AAAI Press, pp. 135-147.

Tyler, S. 1978. *The Said and the Unsaid: Mind, Meaning, and Culture.* New York: Academic Press.

Weitz, R.R. 1990. Technology, Work, and the Organization: The impact of expert systems, *The AI Magazine*, Summer, pp. 62-81.

Wenger, E. 1990. Toward a theory of cultural transparency: Elements of a social discourse of the visible and the invisible. PhD. Dissertation in Information and Computer Science, University of California, Irvine.

Wilkins, D. C., Clancey, W. J., & Buchanan, B. G. 1988. On using and evaluating differential modeling in intelligent tutoring and apprentice learning systems. In J. Psotka, D. Massey, & S. Mutter (eds), *Intelligent Tutoring Systems: Lessons Learned*, Hillsdale, NJ: Lawrence Erlbaum Publishers, pp. 257-284.

Winograd, T. and Flores, F. 1986. Understanding Computers and Cognition: A New Foundation for Design. Norwood: Ablex.

Wynn, E. 1991. Taking Practice Seriously. In J. Greenbaum and M. Kyng (eds), *Design at Work: Cooperative design of computer systems.* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 45-64.

Zuboff, S. 1988. *In the Age of the Smart Machine: The future of work and power.* New York: Basic Books, Inc.