New Perspectives on Working, Learning, and Collaborating and Computational Artifacts in Their Support

Gerhard Fischer
Center for Life-Long Learning and Design (L^{3}D)
Department of Computer Science and Institute of Cognitive Science
University of Colorado at Boulder

Abstract

Human-computer interaction has refocused many research efforts within computer science from a technology-centered view to a user-centered view. Work-centered design transcends user-centered design by acknowledging that not only are most people novice or generic users of computer systems, but skilled workers in specific domains, as well. These efforts need to be augmented by a learner- and collaboration-centered design perspective that emphasizes the dynamic and collaborative, rather than the static and individualistic, nature of human knowledge and work.

The current research efforts of reinventing and reengineering computational environments support the integration of working, learning, and collaboration. New conceptual frameworks *and* computational environments are also needed that will serve as "objects-to-think-with" to demonstrate, communicate, and open up for critiquing the emerging new understanding. In this paper, I explore how new perspectives transcend the dominant current understanding and discuss attempts to turn these perspectives into reality.

1 Introduction

Wisdom is not a product of schooling but the
life-long attempt to acquire it. (A. Einstein)

Learning is a new form of labor [34] and working is often (and needs to be) a collaborative effort among colleagues and peers. In the emerging knowledge society [4], an educated person will be someone who is willing to consider learning as a lifelong process. More and more knowledge, especially advanced knowledge, is acquired well past the age of formal schooling, and in many situations through educational processes that do not center on the traditional school. Seen in this context, working, learning, and collaboration become intimately intertwined rather than being three different and distinct activities.

In our research in human-computer interaction (HCI) over the last fifteen years, we have created conceptual frameworks and innovative systems and have conducted assessment studies to address problems of working, learning, and collaboration with computational artifacts. The content domains of our work are design activities in which design is understood very broadly as the process of determining how things ought to be. Design can be seen as a fundamental activity within all professions [29]. Civil engineers design bridges, lawyers design strategies for cases, politicians design policies, educators design curricula and lessons, and software engineers design software. Design is a collaborative, argumentative process without optimal solutions but with trade-offs. It is impossible for design processes to account for every aspect that might affect the designed artifact. Therefore design must be treated as an evolutionary process in which all stakeholders continue to learn new information and insights as the process unfolds.

The intertwining among learning, working, and collaboration becomes obvious in design, particularly in high-technology fields. Designers are constantly working, learning, and collaborating, which results from a growing recognition that in the information age, change is unavoidable and obsolescence is guaranteed. Learning can no longer be considered a process that occurs only in schools. We need to rethink, reinvent, and reengineer companies, universities, schools, and the relationship among them by exploring new relationships among learning, working, and collaboration.

This article argues for three major issues:

1.) Working, learning, and collaborating need to be integrated, which requires us to move beyond the current boundaries of HCI research perspectives and developments.

2.) Representations are needed that support mutual understanding and allow design stakeholders to work, learn, and collaborate. The design of modern computational environments is characterized by a "symmetry of ignorance" of these stakeholders. None of those involved (e.g., software designer(s), domain designer(s), client(s), customer(s)) as individuals or as one group will have the knowledge necessary for the development of a computational environment.

3.) HCI needs to be viewed as a design science. The HCI community should not be content with reflecting on and evaluating designs developed by other communities, but should itself invent and develop new conceptual frameworks and computational artifacts. To demonstrate our own adherence to this goal, short examples of our system-building efforts are included.

2 Beyond Human-Computer Interaction

In this section, I will briefly summarize some of the major shortcomings of current HCI research and development efforts with respect to the goal of intertwining working, learning, and collaborating (a detailed discussion of these arguments can be found in [9]).

2.1 Human-Computer Interaction Is More than User Interfaces

Applying the Macintosh style to poorly designed applications and machines is like trying to put Béarnaise sauce on a hotdog! (A. Kay)

Human-computer interaction is more than "screen-deep." The interface is important—but if we change only interfaces and not the systems behind them we will only be able to scratch the surface. We should strive for "interfaceless systems" in which nothing stands between users and their tasks (and in which system objects become "ready-at-hand" in a Heideggerian sense). Human-computer interaction should be concerned with tasks, with shared understanding, with explanations, justifications, and argumentation about actions, and not just with interfaces. Although the usual concerns of interface designers (creating more legible types, designing better scroll bars, developing models of keystroke use) are all important, they are secondary considerations. The essential challenges lie in improving the ways people can use computers to work, think, communicate, learn, critique, explain, argue, debate, observe, decide, calculate, simulate, and design. In the future, the emphasis has to be on humans and their tasks—not on computers, interfaces, and tools.

2.2 Make Systems Useful and Usable

If ease of use was the only valid criterion, people would stick to tricycles and never try bicycles. (Engelbart)

Useful computers that are not usable are of little help; but so are usable computers that are not useful [7]. One of the major goals of human-computer interaction research must be to achieve these two goals—usefulness and usability—simultaneously. The "versus" relationship between useful and usable can be turned into an "and" relationship (1) by using familiar representations (based on previous knowledge and analogous situations), (2) by exploiting the strengths of human information processing, (3) by integrating knowledge in the head with knowledge in the world, and (4) by designing "better" systems that take advantage of the unique possibilities of computational media (specifically by focusing on "on demand" notions, such as learning on demand, using on demand, and detail on demand).

High-functionality computer systems (such as Unix, Word, Canvas, and Mathematica) illustrate the tension between usefulness and usablity by creating a "tool-mastery" burden that can outweigh the advantage of the broad functionality offered. Many approaches that have represented major advances in human-computer interaction, such as direct manipulation [18] (bridging the interface gulf by representing the world of the computer as a collection of objects that are directly analogous to objects in the real world), lose some of their power in high-functionality systems in which the complex and abundant functionality can neither be represented explicitly on the screen nor be explored by browsing mechanisms.

2.3 A Broader View of Communication and Collaboration Processes

Intellectual teamwork [16] is an increasingly important part of knowledge workers' activities and innovative computational environments are needed that help communities of practice [19] to perform better. Communication and coordination processes provide a focus for a number of research efforts. A communication and coordination perspective illustrates the requirement to include support for communication with:

- ourselves (e.g., capturing our thoughts of the past, allowing us to create personalized information environments that extend the knowledge we can keep in our head [1,21]),
- our tools (e.g., knowing which tools exist, how they can be used, and how they can be tailored to our specific needs),
- domain knowledge (e.g., knowing the major concepts and strategies of a particular domain),
- our colleagues (e.g., supporting short-term as well as long-term, indirect collaboration [10]),
- other humans (e.g., supporting interdisciplinary computer-supported cooperative work), and
- our agents and critics (e.g., in the context of cooperative and distributed problem-solving systems).

2.4 Support Human Problem-Domain Interaction

Interfaces get into the way. I don't want to focus my energies on an interface. I want to focus on the job. (D. Norman)

To bring tasks to the forefront, computers must become "invisible" [32,33]. If the most important role for computation in the future is to provide people with a powerful medium for expression, then the medium should support them in working on the task rather than require them to focus their intellectual resources on the medium itself. To achieve this goal, human-computer interaction needs to advance to *human problem-domain interaction* [11], requiring that the major abstractions of a given domain are modeled in the computer. This will enable users to describe things briefly because the systems allow them to interact with domain-oriented concepts. To achieve human problem-domain interaction, we have to sacrifice generality for the power of specialized interactions. The domain-oriented design of artifacts supports the grounding of interaction, creates representations for mutual understanding, and allows referential anchoring for working, learning, and collaborating.

Human problem-domain interaction puts owners in charge by allowing them to communicate with the systems at a level that is *situated* within their own world [30]. By supporting representations for mutual understanding [5], such as prototypes, mock-ups, scenarios, images, or visions of the future, human problem-domain interaction makes it easier for owners of problems to participate in the design process because the representations of the evolving artifacts are less abstract and less alienated from practical-use situations. By keeping owners in the loop, human problem-domain interaction supports the integration of problem framing and problem solving [25,26] and allows the ongoing evolution of computational environments. By making information relevant to the task at hand [14], systems are able to deliver relevant knowledge, in the context of a problem or a service, at appropriate moments for consideration by human professionals.

3 Working, Learning, and Collaborating

3.1 Integration of Working and Learning

Learning is part of living, a natural consequence of being alive and in touch with the world, and not a process separate from the rest of life. What learners need, therefore, is not only instruction but *access* to the world (in order to connect the knowledge in their head with the knowledge in the world [21]) and a chance to play a meaningful part in it. Education should be a distributed lifelong process by which one learns material as one needs it. School learning and workplace learning [27] need to be integrated. We refer to workplace learning not as it is currently practiced (i.e., companies imitating school learning by sending their employees to decontextualized classrooms), but as it could be or should be. Examples include apprenticeship-style relationships, such as internships for doctors and Ph.D. students. In such learning situations, problems are not given, but need to be framed. Collaboration is critical and learning is firmly integrated with working. Figure 1 compares school and workplace learning along a number of dimensions.

	Schools	**Workplace**
EMPHASIS ON:	"basic" skills	education embedded in ongoing work activities
POTENTIAL DRAWBACKS:	decontextualized, not situated	important concepts are not encountered
PROBLEMS ARE:	given	constructed
NEW TOPICS:	defined by curricula	arise accidentally from work situations
STRUCTURE:	pedagogic or "logical" structure	work activity
ROLES:	expert-novice model	reciprocal learning
TEACHERS/ TRAINERS:	expound subject matter	engage in work practice
MODE:	instructionism (knowledge absorption)	constructionism (knowledge construction)

Figure 1: A comparison of school and workplace learning

3.2 Motivation

One of the benefits of integrating working and learning is the potential increase in motivation. Motivation to learn new things is critically influenced by optimal flow, a continual feeling of challenge, direct engagement, the right tools for the job, and a focus on the task [3]. Users are willing and motivated to learn when the following conditions hold: (1) they actively desire and control learning, (2) they are successful in finding and using new information, (3) they can see the immediate benefit of learning something new to their current working situation, and (4) their environments are intrinsically motivating and allow them to achieve interesting results with a reasonably small effort.

3.3 Collaboration

Many collaboration technologies (e.g., most CSCW systems) employ the computer as a medium with few interpretable components. Future computational environments need to integrate humans and computational resources more creatively. Computational environments that can interpret objects, actions, and artifacts (not only from a tool perspective, but from a domain perspective) can make information and resources available at the bidding of the user, whereas persons become a skill resource only when they consent to do so and they can also restrict time, place, and methods as they choose. The big expectation of the National Information Infrastructure, namely that Nobel Prize winners will be accessible by every school child, ignores the fact that most Nobel Prize winners will have anything but time to respond to every question directed their way.

To increase the computational support of collaborative environments, a limited shared context must be established. General-purpose information spaces can have only a limited notion of users' tasks at hand. *Domain-oriented design environments* [8] exploit domain semantics and the design context to actively notify designers when there is information they should know. Many current design systems are limited because they function only as "keepers" of the artifact, in which one deposits representations of the artifact being designed. Our experience has shown that designers integrate designing and discussing in such a way as to make separate interpretation difficult [22]. Talking *about* an artifact means talking *within* the context of the artifact (and not in separated e-mail conversations or design rationale handbooks). Later interpretation of discussions requires reconstruction of the context in which they were originally elicited. The most important implication of this view is that design artifacts must not be artificially separated from communication about them.

A new type of collaboration ("grass-roots communities") will be possible through exploiting the resources of virtual communities [23]. Systems such as the world-wide web provide collective expertise (as well as nonsense). The growing megabytes of content are contributed by volunteers and the combination of free expression, lack of central control, many-to-many communication access, and volunteer effort has created a new kind of social organization.

4 Instrumental Versions of Integrated Environments

The integration of working and learning and new approaches toward learning [24] emphasize that learning (1) is a process of knowledge construction, not one of knowledge recording or absorption; (2) is knowledge dependent; and (3) is highly tuned to the situation in which it takes place. This requires computational environments that are *simultaneously* worker/learner-directed and supportive. These requirements are satisfied neither by intelligent tutoring systems nor by interactive learning environments or application programs.

In our research, *domain-oriented design environments* have emerged as systems serving the integration of working, learning, and collaborating by modeling problem domains. They (1) allow users to focus on their tasks (and not just on the interface), (2) increase the usefulness without sacrificing usability, (3) facilitate human problem-domain interaction, and (4) support short-term and indirect, long-term collaboration. In the context of these research efforts, we have explored topics such as design by composition, design by modification, the integration of problem framing and problem solving, the use of critics to increase the back-talk of situations, and the reconceptualization of breakdowns as sources for creativity. *Critiquing* and *proactivity* both support the integration of learning, working, and collaborating, but are founded on different role distributions between designers and computational environments.

4.1 Critiquing

Critics in design environments [10,12,15] are programs that "look over the shoulder" of users as they perform tasks in computational environments and signal breakdowns and offer critiques from time to time. Critics compute their advice by using domain knowledge to examine the actions users perform (e.g., information spaces visited) and the products they create (e.g., constructions and specifications). In critiquing, humans select (partial) goals and communicate some of them to the system, attempt to achieve these goals, and retain control of the interactions. Critics detect potential problems and provide information relevant to the identified problems. Users evaluate the critiques and decide how to respond.

4.2 Proactivity

Proactivity [31] uses a different role distribution between humans and computers and integrates learning with working in a different manner. Proactivity allows designers to delegate certain tasks to the system—and in performing these tasks the system uses knowledge that may be unknown, and yet be of interest to the designer. For designers, work-centered events are the triggers of learning episodes; the system provides them with contextualized information in the process of solving a problem today, which might be relevant for future problem solutions. The relevance of this information is still determined by the user.

ProNet, a proactive domain-oriented design environment [31], will be used as a concrete example. ProNet supports the delegation of specific subtasks and provides tightly coupled linkages between the evolving artifact and the knowledge and argumentation behind the artifact. It provides numerous opportunities to explore and learn task-relevant domain knowledge in the context of the construction of an actual artifact. It was created (1) to support learning on demand; (2) to explore whether a detailed, dynamic, contextualized, user-specified artifact is a more compelling vehicle for learning about new concepts than the "static," uncontextualized, theoretical examples presented in textbooks; (3) to investigate the question whether people will actually take advantage of the computational mechanisms provided; and (4) to transcend the boundaries of pencil and paper technologies and on-line tutorials in which the learner mostly reads or at best does exercises suggested by the computer rather than being engaged in self-directed, authentic activities.

4.3 A Scenario

The following abbreviated scenario (for details, see [31]) is provided to illustrate how proactivity was used by non-expert, inexperienced network designers to learn while working.

Problem Context and Goals. The computer network designers investigated a networking problem based a real networking situation at the University of Colorado Biology Department. In this problem, three existing networks were located in five rooms. The workstations shown were Macintoshes using a proprietary AppleTalk protocol on LocalTalk media, but the networks were considered to be too slow. Figure 2 shows the three networks that were part of the initial problem. All wires in this figure were actually red in color, indicating AppleTalk on LocalTalk cables. The designers tried to achieve the following goals: (1) all 18 computers and the printer located in the five rooms should be able to communicate via the network, (2) devices on all three networks should be able to use the server and printer located in the room in the top center of the design, and (3) the overall network should be faster than the current design, yet easy to install and maintain.

Observed solution sequence and activities. After studying the problem statement and looking over the initial problem state, the following major activities were observed:

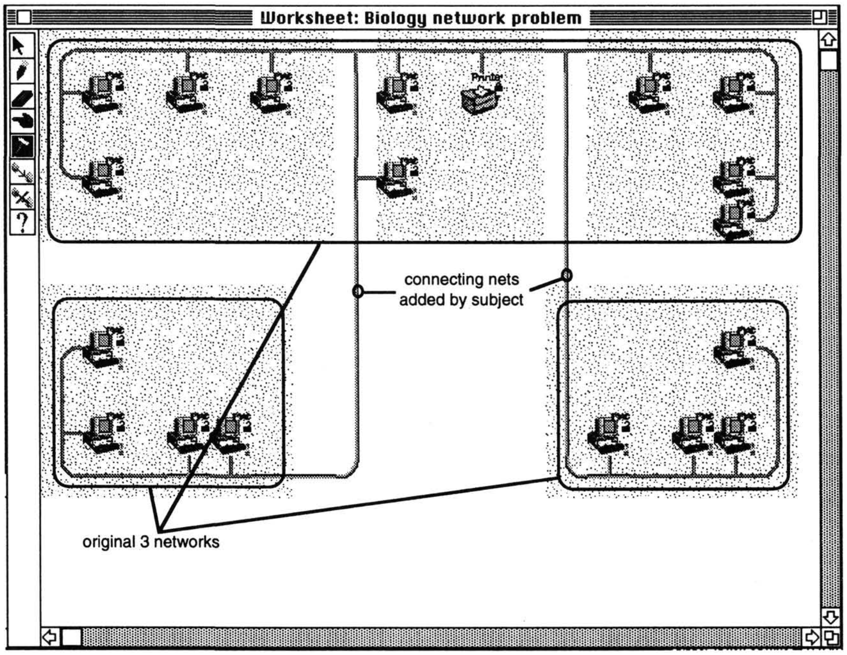

Worksheet: Biology network problem
Printer
connecting nets added by subject
original 3 networks

Figure 2: Problem state after construction tasks to connect the subnets.

1.) *Accessing domain knowledge*: A designer used the domain knowledge hypermedia to look up the word "topology" in the glossary, read about topological design issues for several minutes, remarked, "Good, this is what I needed to know...," and went on to construction tasks.

2.) *Construction activities*: Designers selected the wire item in the gallery and then used the worksheet drawing tool to connect the three subnets and form one large network, as shown in Figure 2.

3.) *Using proactivity to obtain design details*: Designers then enabled proactivity so the system would compute design details using the global priorities. Figure 3 shows the resulting design. Three new details appeared in the upper part of the design. From left to right these are "router," "gateway," and "routing gateway," respectively. All wires in the design changed from red to green, except for one segment of wire between the routing gateway and the gateway where the printer is connected to the network. This one segment remained red.

Learning from unfamiliar design details. Designers noticed this change in network color and requested a local explanation of what the green wire meant by using the

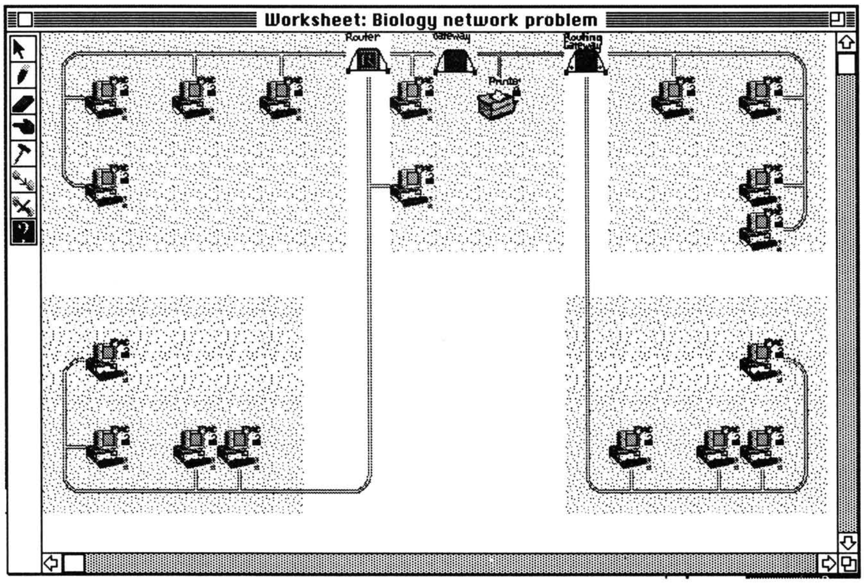
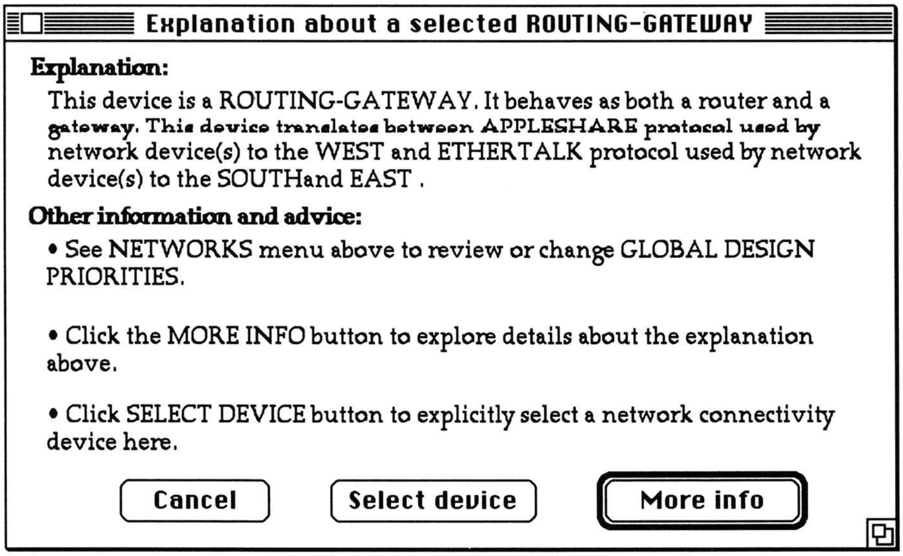


Figure 3: The problem state after enabling proactivity.

query tool labeled with "?". After finding that EtherTalk protocol on Thin-Ethernet media was used, they accessed domain knowledge about the pros and cons of using this protocol and media. They then requested and read a local explanation of the routing gateway that appeared in the top right of the design. Figure 4 shows the local explanation of the device, illustrating the contextualized explanation.

Changing artifact specifications. Designers requested a local explanation for the printer in the design because its icon indicated that it was "locked" on "user-specified priorities." They then selected (in a specification dialog box not shown here) "allow flexibility" for both "how should network requirements be computed next time?" and "current protocol requirements," indicating that the system should proactively compute the best protocol and media using the global design priorities.

Allowing proactivity to obtain new details. After printer specifications were "unlocked," the system proactively computed new network and media requirements for the printer and updated the rest of the design.



Explanation:

This device is a ROUTING-GATEWAY. It behaves as both a router and a gateway. This device translates between APPLESHARE protocol used by network device(s) to the WEST and ETHERTALK protocol used by network device(s) to the SOUTHand EAST .

Other information and advice:

• See NETWORKS menu above to review or change GLOBAL DESIGN PRIORITIES.

• Click the MORE INFO button to explore details about the explanation above.

• Click SELECT DEVICE button to explicitly select a network connectivity device here.

Figure 4: Local explanation of a new routing gateway design detail.

5 Design for Mutual Understanding and Change

If a lion could talk, we could not understand him. (L.Wittgenstein)

Our work on domain-oriented design environments is based on an understanding of design activities as intrinsically collaborative and ongoing. Complexity in design arises from the need to synthesize different perspectives on a problem, the management of large amounts of information potentially relevant to a design task, and understanding the design decisions that have determined the long-term evolution of a designed artifact. Design activities require collaboration among stakeholders, because they are characterized by a *symmetry of ignorance*, meaning that no individual stakeholder (or individual group of stakeholders) knows all the relevant knowledge.

A major challenge of system building is to identify and refine the system requirements of domain workers, and to implement appropriate functionality. Representations in the context of our work are created not primarily for computational processing but (1) to elicit knowledge from domain workers that is often tacit and therefore not easily expressible in abstract situations, and (2) to communicate the intentions and background between system builders and domain workers.

Communication among stakeholders (environment developers, domain designers and clients) is difficult because they use different languages. Explicit representations ground collaborative design by providing a context for communication. Representations help to detect communication breakdowns caused by unfamiliar terminology and tacit background assumptions, and turn the breakdowns into opportunities to create a shared understanding.

5.1 Shared Context in Design Environments

An important component of a shared context is the *intent* of the collaborators. A shared understanding of intent promotes mutual intelligibility by serving as a resource for assessing the relevance of information within the context of collaboration. In our design environments, design activities, including the communication of intent, are centered around artifacts. By capturing the intentions and priorities of designers and associating them with the artifacts, design environments can locate stored artifacts and information relevant to a designer's task at hand and can provide the designer with resources for assessing the relevance of delivered information.

Domain-oriented design environments address these problems in three ways. First, a domain-orientation allows a default intent to be assumed, namely, the creation of a "good" artifact in the given domain. Second, a construction situation (see Figure 2) can be "parsed" by the system, providing the system with information about the artifact under construction. Third, a specification component allows designers to explicitly communicate high-level design intentions to the system, thereby establishing a shared context between the designers and the design environment.

5.2 Seeding, Evolutionary Growth, and Reseeding

Design problems are intrinsically ill-defined, open-ended, and "wicked," making it impossible to predict, let alone collect, all the potentially relevant information in advance. Design environments must capture information continuously over the lifetime of the system [17] and make that information available to designers when it is relevant to their particular tasks. We have developed a process model for the evolution of domain-oriented design environments [13] consisting of three phases: seeding, evolutionary growth, and reseeding (see Figure 5).

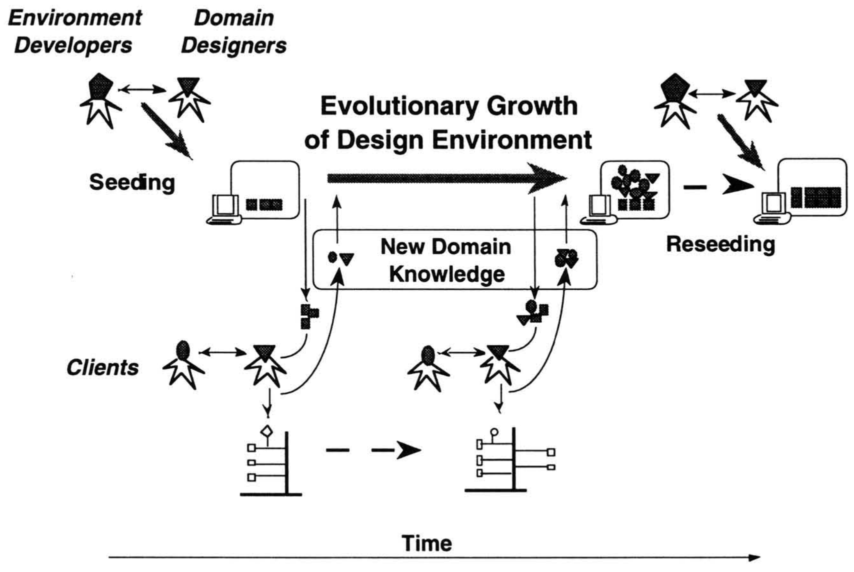


Figure 5: A process model for the development and evolution of domain-oriented design environments

A *seed* is a collection of knowledge and procedures created through a collaboration between environment developers and domain designers. To design new artifacts that are useful for skilled domain workers, either (1) environment developers have to understand the domain concepts and the use activities of the domain designers, (2) domain designers have to understand the possibilities and limitations of computational artifacts, or (3) domain designers must be able to give complete descriptions of their demands, which we know is not possible. Seeds should stimulate, focus, and mediate discussion between the stakeholders, and they should provide mechanisms to capture additional knowledge during the incremental growth phase. There is no absolute requirement for the completeness, correctness, or specificity of the information in the seed. In fact, it is often its shortcomings in these respects that provoke input from designers.

Evolutionary growth during system use is a process of adding information related directly or indirectly to the artifact being designed. Thus, the artifact is the foundation for evolutionary growth. During the growth phase the designers who use the system are primarily focused on their task at hand. Information input is highly situation specific—tied to a specific artifact and stated in particular rather than in general.

Information will grow over time, order will eventually break down, and the system will begin to degrade in its usefulness.

Reseeding is necessary when evolutionary growth stops proceeding smoothly. During reseeding, the system's information is restructured, generalized, and formalized to serve future design tasks. The reseeding process creates a forum to discuss what design information captured in the context of specific design projects should be incorporated into the extended seed to support the next cycle of evolutionary growth and reseeding. Tools contained in design environments support reseeding by making suggestions about how the information can be formalized [28].

5.3 End-User Modifiability

Convivial tools are those which give each person who uses them the greatest opportunity to enrich the environment with the fruits of his or her vision. (I. Illich)

The message derived from the "seeding - evolutionary growth - reseeding" model is that no matter how much software designers try to anticipate and provide for what users will need, the effort always falls short because (1) it is impossible to know in advance what is needed, (2) knowledge is tacit, and (3) the world changes. It is an empirical fact that all successful software undergoes changes.

The approach described by our model documents well how large software systems, such as Symbolics' Genera, Unix, the X Window System, have evolved over time. In such systems, users develop new techniques and extend the functionality of the system to solve problems not anticipated by the system's original authors. New releases of the system often incorporate ideas and code produced by users. In the same way that these software systems are extensible by programmers who use them, design environments need to be extended by domain designers who are neither interested nor trained in the (low-level) details of computational environments [20].

End-users may wish to have functionality that fits their needs, but the creation of this functionality is a difficult task. Two major approaches, namely programmable design environments and collaborative work practices, make end-user programming a more realistic challenge.

Programmable design environments [6] are designed to cope with complexity from different angles by integrating a number of distinct elements: (a) an "application-enriched" programming environment, (b) a "critiquing component" that monitors the user's work and occasionally offers suggestions for changes or tutorial assistance, (c) a "catalog" of illustrative or exemplary work that the user can employ as a starting

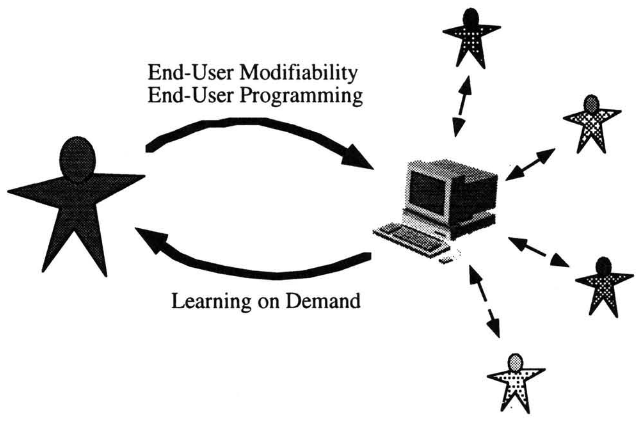


Figure 6: Learning on Demand and End-User Modifiability

point for his or her own work, and (d) embedded tutorial components that the user can access for learning about the application or domain. The first of these elements is primarily aimed at alleviating the problems of complexity faced by the experienced user, whereas the last three of these elements might be viewed collectively as alleviating complexity for the less experienced user.

Collaborative work practices [20] can support end-user computing. In any organization that deals with the same computational artifacts there will eventually be power-users and local developers who will acquire the knowledge to tailor environments to the specific needs of a group. Social resources provided by the community of practice will assist the individual to cope with changes and new demands.

6 Lessons Learned From Our System-Building Efforts

Domain-oriented design environments address the research issue articulated in section 2, "Beyond Human-Computer Interaction": (1) they deal not only with interfaces and tools, but with artifacts and design knowledge of domains; (2) they make an attempt to close the gap between useful and usable by hiding low-level computational details; (3) they support a variety of collaboration processes; and (4) they support human problem-domain interaction. They are instrumental versions of systems that are simultaneously user-directed and computationally supportive.

The critiquing paradigm offers learning opportunities (e.g., by supporting learning on demand [6]) and collaboration opportunities (e.g., by investigating the thinking and the perspectives of other designers as those are embedded in the critiquing itself as well as in argumentation and cases associated with the problem at hand [10]).

Proactivity addresses the production paradox (productive work cannot be done without learning, and learning is prohibited by a lack of time [2]) by helping users get real work done while providing opportunities to learn from the help that is provided. As new details appear in the design, the user can request an explanation of what they are and why they are needed and use these devices to access other related domain knowledge. A shared understanding between designer and system is achieved through the construction and the specification. The construction environment supports experiential cognition allowing the user to create or modify design artifacts and then observe the proactive responses of the system to these actions. Local explanations and argumentative hypermedia allow the designer to reflect on pros and cons, alternatives, and decisions by accessing a web of factual knowledge related to the problem being solved.

7 Human-Computer Interaction: A Design Science

Der Worte sind genug gewechselt,
lasst mich doch endlich Taten sehen. (Goethe, "Faust")

Many researchers and developers in the HCI community have been historically content with evaluating and assessing computational artifacts created by other groups. I strongly believe that HCI needs to become much more a design science, intertwining theory, system-building efforts, and assessment, as illustrated in Figure 7. The HCI community should not only reflect but also propose and develop new designs and new methods and new environments to be used in design. We should not only reflect but enable change. We cannot be content to leaving things as they are or assume that other people should change them.

A design science always has to be prescriptive; it cannot remain only descriptive. Whereas we have to build on as much descriptive knowledge as possible (e.g., strengths and weaknesses of human information processes, properties of technologies, work practices as they exist), we need to decide which kind of computational artifacts we want to have to serve the purposes of working, learning, and collaboration. Our prescriptions will not only be technological, but we have to be (or will implicitly be) social reformers, because information technologies will change how people learn, work, and collaborate. We have to find a careful balance between tradition and transcendence [5]. Designers can prompt and support change in

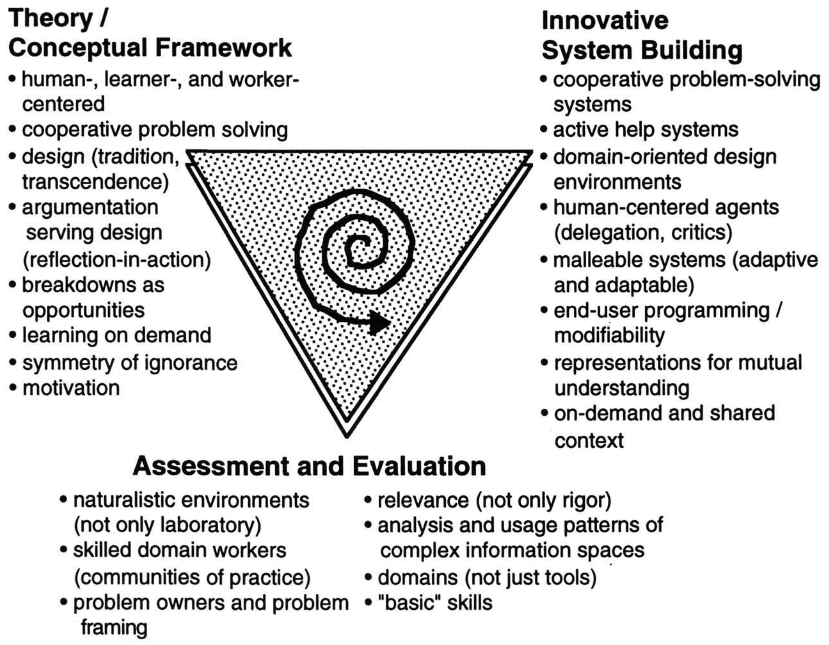

Theory / Conceptual Framework
• human-, learner-, and worker-centered
• cooperative problem solving
• design (tradition, transcendence)
• argumentation serving design (reflection-in-action)
• breakdowns as opportunities
• learning on demand
• symmetry of ignorance
• motivation
Innovative System Building
• cooperative problem-solving systems
• active help systems
• domain-oriented design environments
• human-centered agents (delegation, critics)
• malleable systems (adaptive and adaptable)
• end-user programming / modifiability
• representations for mutual understanding
• on-demand and shared context
Assessment and Evaluation
• naturalistic environments (not only laboratory)
• skilled domain workers (communities of practice)
• problem owners and problem framing
• relevance (not only rigor)
• analysis and usage patterns of complex information spaces
• domains (not just tools)
• "basic" skills

Figure 7: The intertwining between conceptual frameworks/theory, innovative systems, and assessment and evaluation

communities of practice but they cannot and should not predetermine it. Design and use mutually shape one another in iterative, social processes, as indicated in Figures 5 and 6. By seeding a design environment, we not only seed an artifact, but we seed a community of practice—and the community of practice changes in using this artifact.

8 Conclusions

This is not the end. It is not even
the beginning. But it is, perhaps,
the end of the beginning. (W. Churchill)

The dominant HCI research issues of the last decade have been (1) WIMPs (windows, icons, menus, and pointers); (2) an emphasis on interfaces; and (3) a focus on beginners. Important contributions have been made, as can readily be observed by anyone who compares today's interfaces to an ASCII terminal connected to a mainframe computer of ten years ago. These major achievements should not be considered the end, but rather the beginning. As HCI researchers, we—in close collaboration with others—not only have to reinvent and reengineer the computational artifacts and media, but we have to change the underlying processes of working, learning, and collaborating. The major argument behind the current business reengineering debate is that investments in information technology have delivered disappointing results because companies tend to use technologies to mechanize old ways of doing business. The same argument holds for education and collaboration: we use technology as add-on to existing practices, rather than to fundamentally rethink what education and collaboration should be all about in the next century. The old frameworks such as instructionism, curriculum, memorization, and decontextualized learning are not changed by technology itself whether we deal with intelligent tutoring systems, multimedia, or world-wide networks. We have to actively contribute to new frameworks, such as lifelong learning, integration of working and learning, authentic problems, self-directed learning, (intrinsic) motivation, collaborative learning, organizational learning, new content, and new unique properties of computational media.

9 Acknowledgments

The author would like to thank the members of the Center for Lifelong Learning and Design (particularly Hal Eden and Jim Sullivan) at the University of Colorado who have made major contributions to the conceptual framework and systems described in this paper. The research was supported by (1) the National Science Foundation, Grant RED-9253425, (2) the ARPA HCI program, Grant N66001-94-C-6038, (3) Nynex, Science and Technology Center, (4) Software Research Associates (SRA), and (5) PFU. During the academic year 1994/95, the author is supported by the "SEL-Stiftungsprofessor" of the Technical University Darmstadt.

10 References

[1] Bush, V.: As We May Think. In: Atlantic Monthly. 176 (1945) pp. 101-108.

[2] Carroll, J. M.; Rosson, M. B.: Paradox of the Active User. In: Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. J. M. Carroll, (Ed.): The MIT Press: Cambridge, MA. (1987) pp. 80-111.

[3] Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. HarperCollins Publishers: (1990).

[4] Drucker, P. F.: The Age of Social Transformation. In: The Atlantic Monthly. (1994) pp. 53-80.

[5] Ehn, P.: Work-Oriented Design of Computer Artifacts. Almquist & Wiksell International: Stockholm, Sweden, (1988).

[6] Eisenberg, M.; Fischer, G.: Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance. In: Human Factors in Computing Systems, CHI'94 Conference Proceedings. Boston, MA. (1994) pp. 431-437.

[7] Fischer, G.: Making Computers more Useful and more Usable. In: Proceedings of the 2nd International Conference on Human-Computer Interaction (Honolulu, Hawaii). Elsevier Science Publishers: New York. (1987) pp. 97-104.

[8] Fischer, G.: Domain-Oriented Design Environments. In: Automated Software Engineering, Vol. 9, No. 2 (1994) pp 177-203

[9] Fischer, G.: Beyond Human-Computer Interaction. In: Mensch Computer Kommunikation. H.-D. Boecker, W. Glatthaar, T. Strothotte (Eds.), Springer Verlag, Berlin-Heidelberg-New York (1993) pp. 274-287

[10] Fischer, G. et al: Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments. In: Human Computer Interaction, Special Issue on Computer Supported Cooperative Work. 7 (1992) pp. 281-314.

[11] Fischer, G.; Lemke, A. C.: Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication. In: Human-Computer Interaction. 3 (1988) pp. 179-222.

[12] Fischer, G. et al: The Role of Critiquing in Cooperative Problem Solving. In: ACM Transactions on Information Systems. 9 (1991) pp. 123-151.

[13] Fischer, G. et al: Seeding, Evolutionary Growth and Reseeding: Supporting Incremental Development of Design Environments. In: Human Factors in Computing Systems, CHI'94 Conference Proceedings (Boston, MA). (1994) pp. 292-298.

[14] Fischer, G.; Nakakoji, K.: Beyond the Macho Approach of Artificial Intelligence: Empower Human Designers - Do Not Replace Them. In: Knowledge-Based Systems Journal. 5 (1992) pp. 15-30.

[15] Fischer, G. et al: Embedding Critics in Design Environments. In: The Knowledge Engineering Review Journal. Cambridge University Press: 8 (1993) pp. 285-307.

[16] Galegher, P. et al, (Ed.): Intellectual Teamwork. Lawrence Erlbaum Associates: Hillsdale, NJ. (1990).

[17] Henderson, A.; Kyng, M.: There's No Place Like Home: Continuing Design in Use. In: Design at Work: Cooperative Design of Computer Systems. J. Greenbaum and M. Kyng, (Ed.): Lawrence Erlbaum Associates: Hillsdale, NJ. (1991) pp. 219-240.

[18] Hutchins, E. L. et al: Direct Manipulation Interfaces. In: User Centered System Design, New Perspectives on Human-Computer Interaction. D. A. Norman and S. W. Draper, (Eds): Lawrence Erlbaum Associates: Hillsdale, NJ. (1986) pp. 87-124.

[19] Lave, J.; Wenger, E.: Situated Learning. Cambridge University Press: Cambridge, UK, (1991).

[20] Nardi, B. A.: A Small Matter of Programming. The MIT Press: Cambridge, MA, (1993).

[21] Norman, D. A.: Things That Make Us Smart. Addison-Wesley Publishing Company: Reading, MA, (1993).

[22] Reeves, B. N.: The Role of Embedded Communication and Artifact History in Collaborative Design. Ph.D. Thesis: University of Colorado, 1993.

[23] Rheingold, H.: The Virtual Community: Homesteading on the Electronic Frontier. Harper Perennial: (1994).

[24] Resnick, L. B., (Ed.): Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser. Lawrence Erlbaum Associates: Hillsdale, NJ. (1989) .

[25] Rittel, H.: Second-Generation Design Methods. In: Developments in Design Methodology. N. Cross, (Ed.): John Wiley & Sons: New York. (1984) pp. 317-327.

[26] Schoen, D. A.: The Reflective Practitioner: How Professionals Think in Action. Basic Books: New York, (1983).

[27] Scribner, S.; Sachs, P.: On The Job Training: A Case Study. In: National Center on Education and Employment. (1990) pp. 1-4.

[28] Shipman, F.: Supporting Knowledge-Base Evolution with Incremental Formalization. Ph.D. Thesis: University of Colorado, 1993.

[29] Simon, H. A.: Cognitive Science: The Newest Science of the Artificial. In: Perspectives on Cognitive Science. D. A. Norman, (Ed.): Ablex Publishing Corporation, Lawrence Erlbaum Associates: Norwood, NJ - Hillsdale, NJ. (1981).

[30] Suchman, L. A.: Plans and Situated Actions. Cambridge University Press: Cambridge, UK, (1987).

[31] Sullivan, J.: A Proactive Computational Approach for Learning While Working. Ph.D. Thesis: University of Colorado at Boulder, 1994.

[32] Uhlich, E.: Von der Benutzungsoberflaeche zur Arbeitsgestaltung. In: Software-Ergonomie'93. K. H. Roediger, (Ed.): B.G. Teubner: Stuttgart, Germany. (1993) pp. 19-29.

[33] Volpert, W.: Von der Software-Ergonomie zur Informatik. In: Software-Ergonomie'93. K. H. Roediger, (Ed.): B.G. Teubner: Stuttgart, Germany. (1993) pp. 51-65.

[34] Zuboff, S.: In The Age Of The Smart Machine. Basic Books, Inc: New York, (1988).

Gerhard Fischer
Department of Computer Science, Campus Box 430
University of Colorado
Boulder, CO 80309-0430 USA
e-mail: gerhard@cs.colorado.edu