Quintana, C., Krajcik, J., & Soloway, E. (2000). Exploring a Structured Definition for
Learner-Centered Design. In B. Fishman & S. O'Connor-Divelbiss (Eds.),
Fourth International Conference of the Learning Sciences (pp. 256-263). Mahwah, NJ: Erlbaum.

Exploring a Structured Definition for Learner-Centered Design

Chris Quintana, Joseph Krajcik, Elliot Soloway
Center for Highly-Interactive Computing in Education, University of Michigan
1101 Beal Ave., Ann Arbor, MI 48109
Tel: 734-763-6988, Fax: 734-763-1260
Email: quintana@umich.edu

Abstract: As computers become more powerful and prevalent, more researchers are exploring computer tools to help novices gain an understanding into new work domains. The development of such tools involves what has been termed a "learner-centered design" process. But while there is an intuitive notion of what is meant by learner-centered design, there is no formal definition. Here we present our definition of learner-centered design, a definition formed by extending the definition for the traditional "user-centered design" approach. Specifically, we define "learner-centered design" in terms of its target audience, the problems it addresses, and the underlying approach for addressing those problems. From a software development viewpoint, by presenting a more formal definition, we hope to not only better describe what we mean by learner-centered software, but also provide a base by which we can explore more structured design methods for developing and implementing effective learner-centered software.

Keywords: learner-centered design, user-centered design

Introduction

As computers became more powerful and prevalent in the 1980s, the computing community began to realize that the issue of computer usability was becoming more important. This led to the formation of the user-centered design approach (UCD), an approach focusing on computer usability (Norman 1986). Now, software designers are faced with new challenges. For example, Soloway, Guzdial, and Hay (1994) identified a new challenge centered on the use of computers to "support individuals and groups of individuals in developing expertise in their professions, in developing richer and deeper understandings of content and practices." The challenge of using computers to "make people smarter" entails a different, evolving *learner-centered design* (LCD) perspective that considers the needs of *learners*, that is, people who are developing expertise in new and unknown work domains (Soloway et al. 1994).

The notion of designing computer tools centered on the needs of learners has become more popular in recent years. But while there is an intuitive notion of what is meant by "learner-centered design", we need a more concrete definition to facilitate the creation of formal design methods and techniques for software designers building learner-centered software. Here, we present a more structured definition for learner-centered design by reviewing concepts of the traditional UCD approach and extending the UCD framework to describe LCD. We will then briefly review the state of the field and give further research directions for LCD researchers.

A Definition for Learner-Centered Design

We describe learner-centered design by considering three aspects: the audience targeted by each design paradigm, the central problem being addressed by each design paradigm, and the underlying approach each paradigm takes to address the problem. For each aspect, we will review the corresponding UCD ideas put forth by Norman (1986) and extend the UCD description to present an approachable and more structured definition for LCD.

Audience: Users vs. Learners

Software is designed to meet the needs of some target audience who will be using the software. We begin our discussion by defining the target audience for the two design paradigms, defining the audience of "users" for UCD and the audience of "learners" for LCD.

Who Are "Users"?

In descriptions of user-centered design, there is an implicit assumption that the user of a computer tool already possesses some measure of expertise about the work activity they are using the tool to engage in (Norman 1986). Specifically, we distinguish "users" as having the following characteristics:

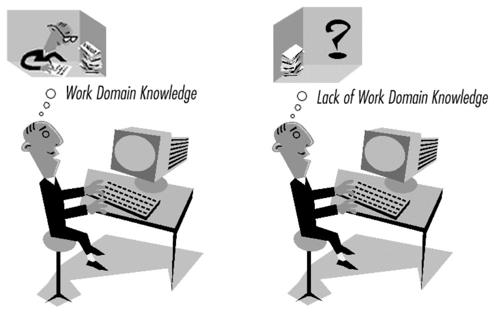
• Users understand the work domain in which they are working and the work tasks they are completing. They simply need a computer tool that will help complete their work tasks easily and efficiently.

• Users engaged in some given work activity share a work culture and therefore can be considered homogenous in a number of important ways (Soloway et al. 1996). The tasks they perform are often similar from user to user, and a designer can rely upon an archetypal user when designing a user-centered tool.

• Users, by the nature of their involvement with their work tasks (e.g., professions, labors of love) have intrinsic motivation for their work and the tool does not have to supply any extra motivating factors (Soloway et al. 1994).

• Users are not necessarily trying to learn about their work through their tools. Rather, they need tools to help them complete their work. Their work domain familiarity allows them to pursue their tasks without any significant growth in the domain. Work professionals will certainly learn new things about their work, but the tools they use will largely stay the same.

In summary, a "user" is knowledgeable and motivated about their work tasks (figure 1a). The goal for the designer is to design a tool to help users complete their work tasks easily and efficiently.



(a) (b)

Figure 1. Target audience: users (a) versus learners (b).

Who Are "Learners"?

Learners will also use a computer system, but in a substantially different manner than "expert users". Whereas users use a tool to complete their work activities, learners are trying to learn about the new work activities through a tool. Thus, in contrast to the "user" definition, we can characterize "learners" as follows (Soloway et al. 1996):

• Learners do not possess the same domain expertise as users. They share neither an understanding of a work domain nor the tasks within it with their professional counterparts.

• Learners are heterogeneous. Learners do not necessarily share a common work culture, background, or understanding, so designers need to consider a larger degree of diversity in background, development, and learning styles in the learner population.

• Learners may not intrinsically motivated in the same manner as experts. Also, since learners lack an understanding of the work domain, they face more obstacles in successfully performing the tasks at hand, which can lower their motivation even more.

• Learners’ understanding grows as they engage in a new work domain. Thus their tools (i.e., their window on the work domain) need to grow and change as they do.

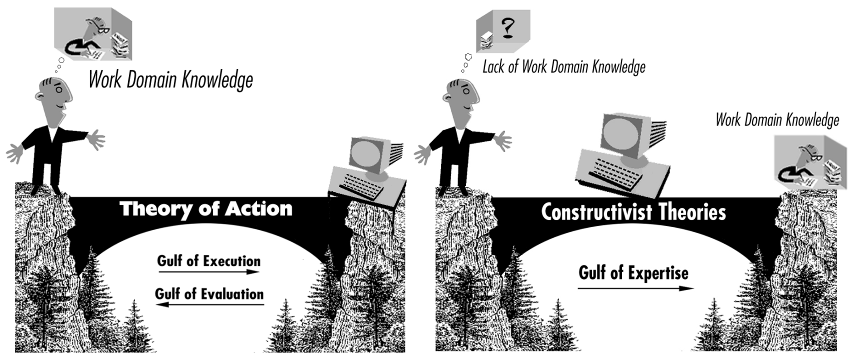
In summary, the focus of UCD is on facilitating task completion by users, whereas the focus of LCD is on building domain understanding in learners (figure 1b). Note however that in describing learners, we are taking a general approach in the definition. We do not make any distinction in the general definition about the age of learners, thus learners could be schoolchildren or corporate workers. The key notion is that a learner will be learning a new work domain. Certainly, the tool design will vary for children or adults and the design team needs to cognizant of the differences. However, for this discussion, we will refer to learners as work domain novices in general.

Problems Addressed: Using Tools vs. Learning Work

Given these descriptions of the target audiences for UCD and LCD, we now describe the central problem being addressed by each design paradigm. In discussing the design problems being addressed, the key point is that UCD addresses the conceptual gap between the user and the tool, while LCD addresses the conceptual gap between the learner and a work domain.

UCD Problem: The Gaps Between User And Computer

Given the UCD goal of designing a usable system, we need to consider what is meant by “usable”. Norman (1986) states that when users use a tool to complete their work, they will have goals in mind that they need to translate into actions to execute on the tool. Once users have executed an action, they must evaluate the tool’s resulting state and interpret that state in terms of their goals. Thus there exist two important discrepancies between the goals of the user and the physical tool. These discrepancies can be represented as “gulfs” between user and system: a gulf of *execution* and a gulf of *evaluation* (figure 2a).



(a) (b)

Figure 2. Problems addressed: UCD gulfs (a) versus LCD gulfs (b).

The gulf of execution is the difference between the goals and intentions of the user and the permissible actions on the tool. The gulf of evaluation reflects the amount of effort the user must exert to interpret the physical tool state (Norman 1990). The “size” of the gulfs corresponds to the difficulties users face in understanding and using the tool to complete their tasks. A usable (or user-centered) system must minimize both gulfs, bringing the user and the tool closer together.

LCD Problem: The Gap Between Learner And Work Domain

With UCD, the designer designs a system to help users bridge the usability gulfs of execution and evaluation. But with LCD, there is an additional gulf to consider. Since learners need to develop an understanding of a work domain, a learner-centered tool needs to support the learner in bridging a gulf of *expertise* between the learner and the work domain (figure 2b).

For a learner to be able to participate in some work domain, the learner needs to understand what kind of activities are in the work domain, facts about the work domain, and knowledge for completing the various activities in the domain. The “size” of the gulf of expertise is proportional to the amount of expertise the learner needs to gain in order to work successfully within the given domain.

Underlying Approach: Supporting Action vs. Supporting Learning

Given the design problems addressed by each design paradigm, we now look at the underlying theoretical approaches used to address the design problems. Specifically, we will see that UCD utilizes a “theory of action” describing how people work with tools, while LCD utilizes learning theories to describe how people gain expertise.

Bridging the Tool-Usability Gap: Designing for Users

One way of reducing the usability gulfs is to bring the system closer to the user through proper design. In other words, designers should design software that bridges the usability gulfs by making execution and evaluation straightforward. In order to meet these design goals, Norman (1986) describes the following conceptual models used in UCD (Norman 1986) (figure 3):

• The designer model is the system conceptualization held by the designer and developed by understanding the user’s tasks, requirements, and capabilities. Note that this should not imply that UCD involves a sole designer working in a vacuum. Task analysis or interviews with domain experts inform the designer model, but the designer model is the central model defined for UCD.

• The system image is the actual physical system.

• The user model is the system conceptualization the user develops to explain to themselves how the system works. Users develop a user model through initial and continued system use.

The designer creates the system image based upon their design model. Given the system image, the user begins to create a user model of the system. If the user model is consistent with both the designer model and the user’s own work understanding, then the system is considered “usable”.

So how can a designer envision a usable and understandable designer model? By understanding how people work with tools. Norman (1986) postulates that in order to develop a design process that realizes consistent conceptual models, designers need a “theory of action” explaining “how people generally do things to complete a task”. Such a theory can guide a designer in building a system that helps users bridge the usability gulfs (figure 2). Norman’s theory of action is expressed as a series of execution steps (establish a goal, an intention to achieve the goal, a specific sequence to meet the intention; execute the specific action sequence) and evaluation steps (perceive, interpret, and evaluate the tool state) (Norman 1986).

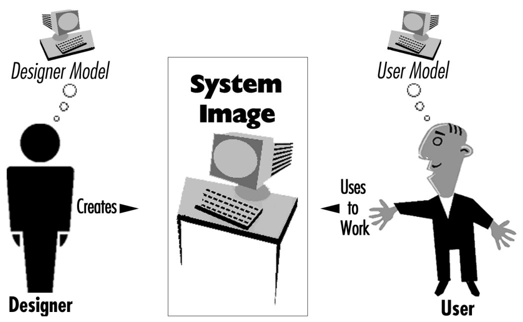


Figure 3. Conceptual models in UCD.

What are the ramifications of such a “theory of action” for designers of user-centered tools? Essentially, a theory of action informs the development of UCD heuristics. For example, designers should simplify the tool structure, use affordances and metaphors, make permissible actions and the tool state visible, etc. (Norman 1990, Norman 1986). By analyzing work tasks and understanding how people perform those tasks, designers can design systems that users can operate and understand to successfully engage in their work activities.

Bridging The Domain-Learnability Gap: Designing For Learners

LCD involves additional conceptual models because of the need to design the system image to reflect work domain knowledge in a way that learners can understand. First, the system should reflect a *domain model*, i.e., an expression of the work domain learners are trying to understand. Second, the presentation of domain knowledge must be guided by an *educational model*, which informs how the domain model should be presented to best facilitate the learner's construction of new understanding. These elements impact the designer model of the system. Because of these new conceptual models, a LCD team needs domain experts (e.g., work professionals) and educational experts (e.g., teachers). The designer must design a usable system and realize the contributions of the domain and educational experts (figure 4).

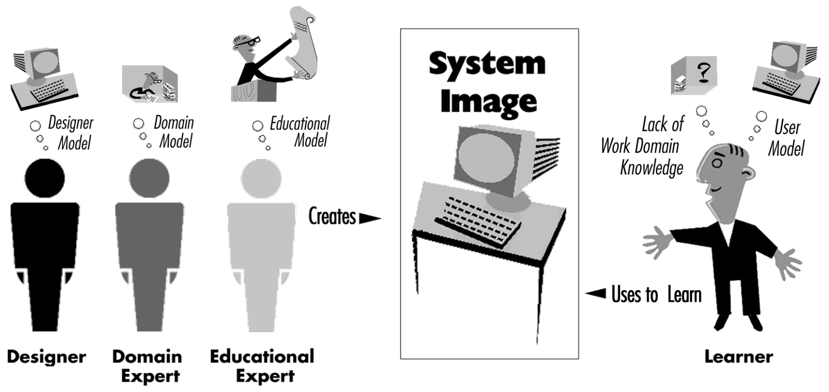


Figure 4. Conceptual models in LCD.

For LCD, the system image needs to incorporate a domain model. Furthermore, the domain model should be presented in a way that learners can begin to build an understanding of the work domain (i.e., through the use of an educational model.) LCD requires theories that describe what we mean by "gaining expertise" to help develop learner-centered design principles. For this, we consider *constructivist* and *social constructivist* learning theories to help us see how to help learners bridge the gulf of expertise (Soloway et al. 1996, Soloway et al. 1994) (figure 2b):

• Constructivist learning theory states that understanding and learning involves active, constructive, generative processes (e.g., assimilation, augmentation, and self-reorganization) (Piaget 1954, Papert 1993). Learning is not a passive information transfer process from expert to novice, but rather an active process that employs a "learning by doing" approach where the learner must manipulate the material they are learning (Soloway et al. 1996).

• Social constructivist learning theory states that learning is enculturation. That is, learning does not occur in a vacuum, but must occur within some representation of the work context so that learners develop an understanding of that work context and culture. Brown, Collins, and Deguid (1996) state that "knowledge is…in part a product of the activity, context, and culture in which it is developed and used". Learners must build their knowledge by participating in the professional work culture to develop an understanding of the common practices, languages, tools, and values of a professional culture (Soloway et al. 1996).

How do these learning theories inform LCD? In essence, learners need to *actively* engage in authentic activities from the work domain. Learning needs to be an active process, where learners ask questions, collect and organize information, and assess their work, all by using tools and engaging in activities from the given work domain. By doing so, learners can begin to develop an understanding of the work culture, i.e., the practices and language of the work domain.

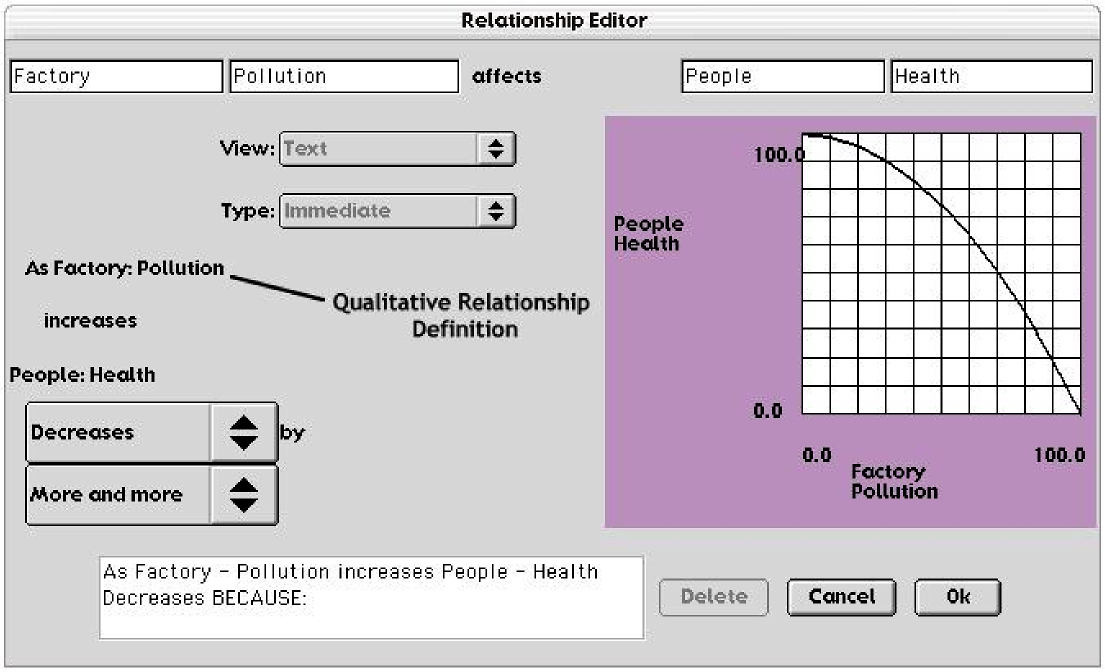
Learners, then, need tools that help them with this active engagement in a work domain. Since learners lack an understanding of the work domain, they need additional support (or *scaffolding*) in their tools to help them engage in the new work activities. Certainly learners cannot use the same tools that domain experts use (i.e., user-centered tools) because of the difference in their levels of expertise. Thus, a learner-centered designer must design tools modeled on expert tools, but structured in ways that allow learners to participate in activities similar to those of domain experts to mediate the learner's development (Belamy 1996).

Just as a theory of action in UCD informs the development of heuristics for building user-centered tools, we are now beginning to distill LCD heuristics from the use of constructivist and social constructivst theories. Many researchers are developing learner-centered tools. For example, in two issues of *Communications of the ACM* (1996a, 1996b) illustrate a variety of learner-centered tools from different researchers. Here, we briefly discuss two LCD heuristics using software examples from our research group's work to ground the discussion on learning theory in LCD. Note, however, that as more learner-centered tools are being developed, a wider range of LCD heuristics and scaffolding strategies are coming to light (e.g., Squires and Preece (1999) present some LCD heuristics and a review of socio-constructivist principles in software design).

Make Complex Tasks Accessible

One tenet of the constructivist approach is that learning occurs from active engagement in a task. However, novice learners need tools that make complex tasks accessible to them. Tasks need to be made accessible, but in a way that does not oversimplify the task to a point where it is contrived or not authentic. One example of making complex tasks accessible is seen in ModelBuilder, a system dynamics modeling tool for science students (Soloway et al. 1996). Students use ModelBuilder to build models of complex systems, such as ecosystems, the atmosphere, etc., in order to explore scenarios and explain the behavior of such complex systems.

One complex task in building system models is defining how different factors in the model relate to each other. For example, in an air quality model, students may want to relate car exhaust, pollution levels, wind speed, and citizens' health. Professional modeling programs can require relationships to be defined by writing differential equations, a task much too difficult for students. ModelBuilder instead makes relationship definition accessible through a relationship editor (figure 5) that allows students to define relationships by *building qualitative sentences* that describe the relationship.



As Factory - Pollution increases People - Health Decreases BECAUSE:

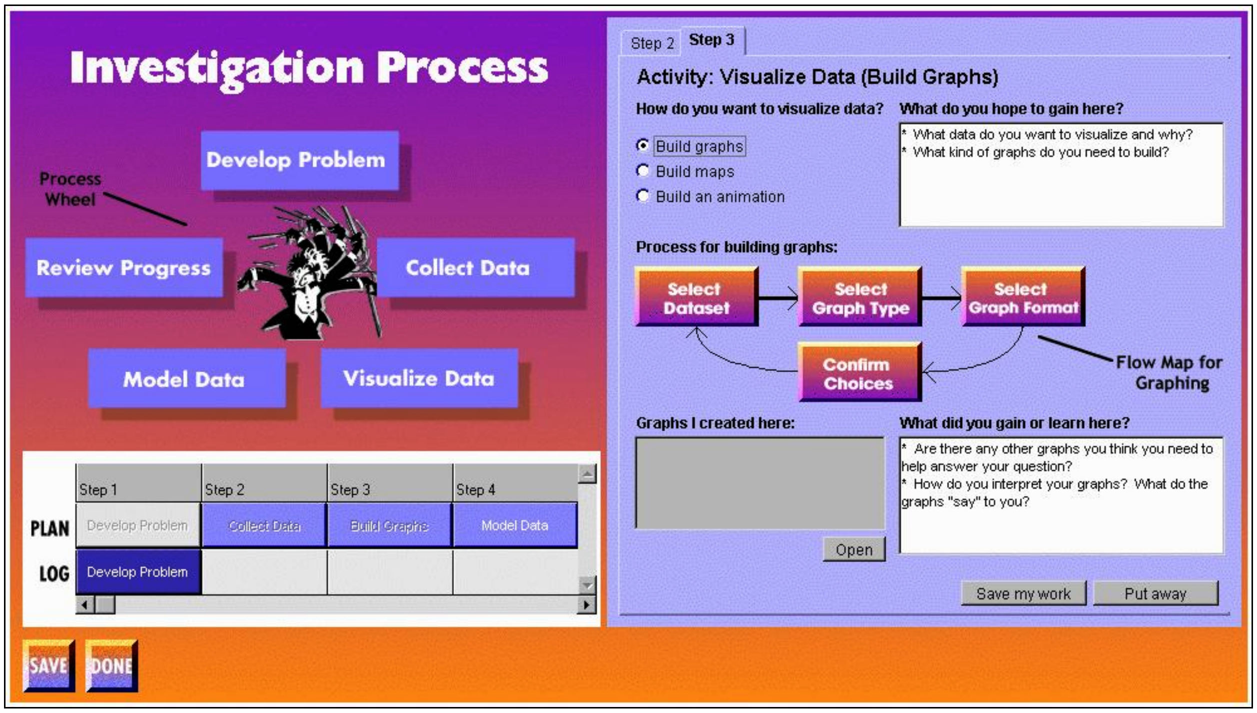
Figure 5. ModelBuilder Relationship Editor

Here, students building an air quality model describe how pollution and health are related. By filling in the sentence, students define the relationship by saying "as pollution increases, people's health decreases." When describing how health decreases, students are presented with qualitative choices (e.g., a little, more and more, the same, a lot, etc.) that they understand, but that are still useful for defining a working model. Students still have to think about the relationships in the model that they are building. However, rather than oversimplifying the task, learners can now build relationships using a tool that is simpler, but that still requires them to reflect on how they believe factors in their model are related.

Visualizing Work Processes and Activities

One tenet of the social constructivist approach is that learning (i.e., enculturation) occurs when learners work within a context that allows them to see and understand the professional work culture. Thus, learner-centered tools should visualize work processes and activities. In other words, tools should reflect a professional work context in such a way that the learner can engage in and understand of the activities, language, practices, etc., that comprise the new work domain. Examples of work process visualization are found in Symphony (figure 6), a scaffolded integrated tool environment that integrates a range of data collection, visualization, and modeling tools within a framework for conducting scientific investigations (Quintana et al. 1999). Students use Symphony to plan an investigation, carry out a variety of science activities, and review the artifacts they have created to assess their investigation and present their results.

Symphony uses a variety of process maps to visualize both the range of possible activities that make up a science investigation and the procedures for carrying out certain science activities. Figure 6 shows the Symphony main screen with two process maps visible. First, on the left is the central *process wheel* that illustrates to students the space of activities involved in a science investigation. Students can survey the process wheel in order to take activities from the map for their plan. By showing students the space of possible activities, they can begin to see and understand what kinds of activities comprise the work of science inquiry.



Activity: Visualize Data (Build Graphs)

How do you want to visualize data?

- (●) Build graphs
- ( ) Build maps
- ( ) Build an animation

What do you hope to gain here?

* What data do you want to visualize and why?
* What kind of graphs do you need to build?

Process for building graphs:

Graphs I created here:

What did you gain or learn here?

* Are there any other graphs you think you need to help answer your question?
* How do you interpret your graphs? What do the graphs "say" to you?

Figure 6. Symphony Process Maps

Second, on the right of figure 6 is a directed *flow map* that describes the procedure for performing certain science activities. The flow map shown in figure 6 specifically illustrates the procedure for building a graph. Students step through the flow diagram to build a graph by pressing the buttons in the map, which in turn launch the appropriate portion of the graphing tool. Not only does the flow map make the graph-building task accessible to the student, it also explicitly presents the steps of the process so that the student can internalize the work involved in graph-building and eventually build graphs without needing the flow map.

These are brief examples of LCD heuristics. Other examples of LCD heuristics include: showing real-world representations, supporting metacognition and reflection, providing multiple representations of data and information, designing task-oriented interfaces, etc. (Soloway et al. 1996, Squires and Preece 1999). By using these and other LCD heuristics and strategies, designers can develop tools that support learners in doing and seeing complex work practices that would be otherwise out of their reach. Such tools differ from past efforts to design software tools for learning. Computer-aided instruction (CAI) (based on behaviorist principles) and intelligent-tutoring systems (ITS) (based on information-processing psychology) attempt to package educational components that can train learners. However, these approaches are more passive in nature, presenting a situation where the computer tools transfer the new information to the learner. Certainly, there are aspects of CAI and ITS that could be incorporated into learner-centered tools. However, their overall approach is too passive in nature. As Squires and Preece (1999) note, there is now a shift of emphasis from behaviorist approaches to more constructivist approaches in order to allow learners to engage in more open-ended, exploratory, and personally meaningful construction of knowledge. Learner-centered tools must then support a "learning by doing" approach to scaffold and allow such active participation by the learner in the culture and practices of the given work domain.

Concluding Remarks and Future Directions

More researchers are exploring the development of learner-centered software. But while there is an ad-hoc perception of what it means to be "learner-centered", we want a more concrete definition for learner-centered design. Here, we described our approach at formulating such a definition. By using the definition of the traditional user-centered design as a base, we hope to begin a dialogue to refine and expand a LCD definition.

From a software development viewpoint, having a more concrete definition for LCD as a base can help us explore and detail further directions for LCD. Once we can describe what we mean by "learner-centered software", we can describe more structured design methods and techniques for LCD. Future research directions include: defining analysis techniques to observe work practices and develop domain models for design, formalizing methods for determining learner needs within a given work domain, categorizing and cataloguing scaffolding strategies and LCD principles to share with the design community, and determining structured methods for assessing the effectiveness for learner-centered software

References

Belamy, R.K.E. (1996) "Designing Educational Technology: Computer-Mediated Change". In B.A. Nardi (ed.), *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press.

Brown, J. S., Collins, A., Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 18, 32-42

Communications of the ACM (1996). Vol. 39, No. 4, pp. 24-49.

Communications of the ACM (1996). Vol. 39, No. 8, pp. 83-109.

Norman, D.A. (1990) *The Design of Everyday Things*, Doubleday-Currency.

Norman, D.A. (1986) "Cognitive Engineering". In D.A. Norman & S.W. Draper (Eds.), *User Centered System Design*. Lawrenece Erlbaum Associates.

Papert, S. (1993) The Children's Machine: Rethinking School in the Age of the Computer, Basic Books, New York.

Piaget, J. (1954) *The Construction of Reality in the Child*. Basic Books, New York.

Quintana, C., Eng, J., Carra, A., Wu, H., Soloway, E. (1999) Symphony: A Case Study in Extending Learner-Centered Design Through Process Space Analysis. *Human Factors in Computing Systems: CHI '99 Conference Proceedings* (Pittsburgh, May) ACM Press.

Soloway, E., Guzdial, M., & Hay, K.H. (1994) "Learner-Centered Design: The Challenge for HCI in the 21st Century". *Interactions*, Vol. 1, No. 2.

Soloway, E., Jackson, S.L., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S.J., Studer, S., Eng, J., and Scala, N. (1996) "Learning Theory in Practice: Case Studies in Learner-Centered Design". *Human Factors in Computing Systems: CHI '96 Conference Proceedings* (Vancouver, Canada, April). Addison-Wesley.

Squires, D., and Preece, J. (1999) "Predicting Quality in Educational Software: Evaluating for Learning, Usability and the Synergy Between Them". *Interacting with Computers*, 11, 467-483.

Acknowledgements

The authors would like to thank the members of the Center for Highly Interactive Computing in Education for their feedback and suggestions, Yvonne Pappas for the artwork, and John Carroll at Virginia Tech for additional comments. This work is supported by the National Science Foundation.