Bibliography

Bransford, J. (2000). *How people learn: Brain, mind, experience and school.* Washington, DC: National Academy Press.

Brown, J. S., Collins, A., & Duigud, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*(1), 32–42.

Chung, M., & DiGiano, C. (2002). A primer on reuse. *Journal of Online Mathematics and its Applications (JOMA)*, 2(1). Retrieved January 29, 2008, from http://mathdl.maa.org/mathDL/4/?pa=content&sa=viewDocument&nodeId=472.

Fincher, S., Petre, M., & Clark, M. (2001). *Computer science project work: Principles and pragmatics.* Berlin: Springer-Verlag.

Garrison, M. M., & Christakis, D. A., & Kaiser Family Foundation. (2005, December). *A teacher in the living room? Educational media for babies, toddlers, and preschoolers.* Retrieved February 14, 2008, from http://www.kff.org/entmedia/upload/7427.pdf.

Jones, S. (2003). *Let the games begin: Gaming technology and entertainment among college students.* Washington, DC. Pew Internet and American Life Project Retrieved February 14, 2008, from http://www.pewinternet.org/pdfs/PIP_College_Gaming_Reporta.pdf.

Kahn, K. (1999). Does easy do it?: Children, games, and learning. Roundtable discussion held at GDC99 on March 16, 17, and 18, 1999. Retrieved May 31, 2006, from http://www.toontalk.com/English/easydoit.htm.

Lave, J., & Wenger, E. (1991). *Situated learning.* Cambridge: Cambridge University Press.

Lenhart, A., Madden, M., Rankin Macgill, A. R., & Smith, A. (2007). *Teens and social media.* Washington, DC: Pew Internet and American Life Project.

National Public Radio. (2005). *Survey shows widespread enthusiasm for high technology.* Retrieved February 14, 2008, from http://www.npr.org/programs/specials/poll/technology.

Pea, R. (1993). Practices of distributed intelligence and designs for education. In G. Solomon (Ed.), *Distributed cognitions: Psychological and ethical considerations.* London: Cambridge University Press.

Pogue, D. (2005). *Demand for educational software drops.* "Morning Edition," National Public Radio, November 14. Retrieved June 1, 2006, from http://www.npr.org/templates/story/story.php?storyId=5011336.

Smerdon, B., Cronen, S., Lanahan, L., Anderson, J., Iannotti, N., & Angeles, J. (2000). *Teachers' tools for the 21st century: A report on teachers' use of technology.* Washington, DC: U.S. Department of Education, National Center for Education Statistics..

U.S. Census Bureau. (2007). *Facts for features, back to school: 2006–2007,* August 16, CB06-FF.11-2. Retrieved January 29, 2008, from http://www.census.gov/Press-Release/www/releases/archives/facts_for_features_special_editions/007108.html.

Chapter 2

What Is Design Knowledge and How Do We Teach It?

Christopher Hoadley and Charlie Cox

Introduction

> Engineering, medicine, business, architecture, and painting are concerned not with the necessary but with the contingent—not with how things are but with how they might be—in short, with design.
> (Simon, 1969, p. xi)

How do we produce the next generation of learning technology designers? To answer that question, we need to know what design knowledge is and how it differs from other types of knowledge. Most important of all, we need to understand how learners acquire it. This chapter draws from the field of design research, which attempts to systematically study design methods and their outcomes as a social science. The psychological or sociological theories of design researchers are also sometimes called *design studies.*[1]

Conveying design knowledge is particularly challenging because though experts indisputably "know" the subject, they often have great difficulty explicating what they know for novices/apprentices. One can describe ad nauseam what some of the characteristics of Frank Lloyd Wright's work are that make it great, but this description is not enough to duplicate his greatness, even for experienced architects, much less novices to the domain. The paradox of teaching design is that designers know things, but they can't tell others about them in a way that novices will understand. In other words, this stuff can't simply be written down and told to people, and *voila!* they become experts. We also have to recognize that experts are unique—they don't know the same things in the same way. We need, then, to get a better grip on what experienced designers know—in whatever sense of the word—and come up with effective, reproducible ways of getting novices to a similar stage, such that they understand the general ideas that all expert designers share, and develop their own unique ways of understanding and applying those ideas.

Why Learning Technology Design is Not a Solved Problem

Design is an important class of human activity because it links theory and practice, bridging scientific activities with creative ones in order to deal with ill-structured, open-ended problems. Furthermore, there are nearly as many design methods as there are design problems. Goel has argued that design thinking is, in fact, a completely distinctive cognitive mode that challenges the characterization of human information processing and cognitive psychology as derived from more workaday problem solving (Goel, 1995).

Good design consists of elegantly managing the complexity of ill-structured, open-ended problems such as those found in the creation of learning technologies. Designers are masters of creating processes to deal with underspecification and uncertainty and at dealing with the complexity of emergent phenomena. They are good at negotiating between what is known in general and the particulars of an individual context or setting. Design knowledge is often meta-knowledge, in that it may lean less toward "answers" and more toward "methods leading to answers."

What, then, can we say designers actually *know*? In the design literature, there are only two nearly universally held principles. First, good design is iterative. Second, iterations only help if some feedback (data) is used to improve the design for the next iteration. In visual arts, the only feedback may be the artist's own conception of the work; in learning technology design, usually feedback includes some form of student assessment, classroom testing, or other data collection. Novice designers are usually taught these two principles. But even these seemingly "universal" principles aren't always (or even usually) put into practice, although nearly all designers believe they should be (Gould & Lewis, 1985).

This disjunction between theory and practice underscores the fact that the question is not answerable in the same way as the question, "What do engineers know?" It would be most useful if, as in engineering, there was usually a "right" (or at least optimal) answer to a given design problem based on predictive models and controlled experiments. We could then give simple tests to see whether designers had learned what is required for practicing their profession. But attempts to solve human problems with the methods of engineering and science often fail in open-ended domains such as planning (Rittel & Webber, 1973). Rittel and Webber identified planning as a "wicked problem": a problem that, owing to human complexities, is not only hard but impossible to answer authoritatively. That is because we have inherently incomplete knowledge about the variables of human behavior. Furthermore, it is impossible—not just difficult, but impossible—to design experiments that control for all of the

relevant features in a human situation. In short, there is no one recipe for using technology to solve educational problems.

An example of the disjunction between theory and practice was examined by Borenstein (1991), in his chronicling of the user interface (UI) design of the "Andrew" project at Carnegie Mellon University—one of the first graphical interfaces ever developed. Borenstein noted a mismatch between software engineering design as it had been taught to him in school and the intrinsic uncertainties and ad hoc processes demanded by acting "as if people mattered." He ultimately decided that using software engineering techniques as a way to design user interfaces is a "noble delusion" (Borenstein).

Fortunately, computer science and engineering departments have become more friendly toward "designerly" ways of thinking. They have realized that design must take unanticipated consequences into account. Unintended consequences are managed via iterative design, in a cycle that Donald Schön has termed "see-move-see" (Schön, 1992). Indeed, cognitive scientists have begun to explore ways in which people externalize their cognition and think *in* and *with* the world instead of merely *about* it (Hutchins, 1995; Kirsh & Maglio, 1994) in ways that echo the traditions of ecological psychology. Indeed, this need to be responsive to unanticipated consequences is not only a challenge but a strength. Schön's "see-move-see" examples show how response to the environment can yield inspiration as well as frustration. In the hands of a skilled designer, surprise outcomes fuel the work, just as variation or even mistakes can inspire jazz musicians in the process of group improvisation.

An example from our own research helps illustrate "see-move-see." In the late 1990s, during the emergence of the Web, we developed Internet-based applications to support science learning (Hoadley, 2002). Even though our software and activities underwent relatively minor changes during the research project, the social context of the technology changed rapidly around us. Initially, when we piloted our activities, we had to explain to students what blue, underlined text signified and how to use forward and back buttons in a browser. In less than three years, students came to class not only with Web skills but with preconceived notions about how to conduct themselves online and what the Internet was "for" (typically perceived to be entertainment and socializing). Though one might try to model this as "prior knowledge" and control for it, the reality is that it is easier to conceptualize as an aspect of the social context of the activity.

Therefore, identifying a single, reliable method for learning technology design amounts to a futile search for the Holy Grail. Flyvbjerg, in his book *Making Social Science Matter* (2001), points out that the Greeks distinguished between three different types and purposes of knowledge.

Episteme corresponds roughly to our notion of science—deduction performed by testing ideas about the world against empirical experience. *Techne*, "craft," refers to the knowledge-in-practice of how to accomplish things, such as carving skills. *Phronesis* corresponds roughly to the exploration of what *should* be, or the setting of values. The successful creation and understanding of learning technology requires a careful balance between all three of these knowledge types. Problem finding and refinement requires *phronesis*, the production of iterations requires *techne*, and the interpretation and incorporation of data requires *episteme*. This is in stark contrast to, for instance, science courses that are usually heavy on episteme or vocational courses that are usually heavy on *techne*. Design requires all three.

An example of such a blending of knowledge is found in the German Bauhaus movement from the early twentieth century (Whitford, 1984). The Bauhaus attempted to combine craft work such as smithing, glassmaking, weaving, pottery, and the like with fine arts such as painting, sculpture, and so on. Additionally, the Bauhaus emphasized architecture and new manufacturing processes and materials; for instance, the Bauhaus produced the first metal tubular furniture. In combining *phronesis* with *techne* and *episteme*, they forged new models for what it means to do design, and their work touched everything from architecture to product design to theories of color and form. In doing so, they had to not only create a space for experts but found a school based on different ways of teaching. Bauhaus students were not just taught to memorize prior works as would have been traditional in an art history course, nor were they schooled in technique in the rigid way that would have been prevalent in craft education of the time. Rather, they learned methods that integrated theory and practice.

What a Designer Knows

Though we cannot specify exactly what is in the head of professional designers of learning technologies, we can group methods into useful categories. In this section, we propose the following classification of design knowledge.[2] It consists of the following categories:

- stages
- values
- roles
- principles
- patterns
- techniques
- design psychology.

We believe that this list covers what is known and conveyed in many design disciplines, not just learning technologies, especially the portions that are unique to design. This list helps us think about or research what designers know, but it also helps teachers of design ensure they make each of these kinds of thinking visible to students (Collins, Brown, & Holum, 1991).

Stages

Stages are ways to structure in time how design progresses through successive iteration and comprehensiveness of detail. This is a chronological view of design in which different phases of the design process contain different types of activities. Stages are an excellent way to describe a design process when certain activities occur primarily at different times in the process. Stages are less useful when the activities in the design process substantially overlap or when there are activities that may occur at any time in the design process. Much of the early literature on design emphasizes the stages view of design methods. In this literature, design is characterized as a process that starts with vague ideas about the possible solutions (and often, vague ideas about the nature of the problem) and progresses to more concrete solutions to the problem. The duration of a "stage" may vary—Schön's "see-move-see" places this in a moment-to-moment time frame, where other stage models such as "prototype-evaluate-prototype" might have stages on the order of weeks or months. A typical sketch of a design process is offered by Moran and Carroll (1996). Their sketch of design stages runs as follows:

1 requirements phase
2 design phase (or specification phase)
3 building phase (sometimes called an *implementation* phase)
4 deployment phase (sometimes called a *dissemination* phase)
5 maintenance phase
6 redesign phase.

These phases proceed iteratively for each "release" of the design. At least a dozen authors have variations of this basic theme with four to ten stages that portray design linearly as a progression from the less determined exploratory work to the more constrained final production of designs. For instance, Kelley reduces this to three stages (Kelley & Littman, 2001), whereas Dick and Carey use five (Dick, Carey, & Carey, 2001). Many more recent design perspectives have criticized the so-called waterfall design process and have attempted to use more complex, nonlinear processes. Moran and Carroll, like many others, do allow for a nonlinear

process, in that there is gatekeeping at each step; if the evaluation of the stage or step shows the work lacking, the designer backs up until they can successfully progress to the next stage. Others describe stages proceeding repeatedly in a "spiral" process (Boehm, 1988) or something less linear. For instance, the star model of design process (Hartson & Hix, 1989) puts evaluation as the central activity in a design process, and the designer bounces from this to other activities in any order but always coming back to evaluation.

Values

Values describe another form of design knowledge. A value might be a social goal, or it might be some quality, such as "simplicity" (Maeda, 2006). When push comes to shove, designers must make choices. Designers make these choices according to their goals or their values in the design process. These values may not yield a specific chronological progression of stages but instead may manifest in a stance that is taken in all the activities in design. Rather than offer detailed examples of how subscribing to a particular value impacts design practice, we want to stress the idea of a value as a "first principle" from which different types of activities are derived. In educational designs, this is sometimes called an *educational philosophy*, reflecting that the goals or values espoused by the designers are more of a stance or sensibility than a simple design constraint or a recipe of any sort for how to achieve the desired outcomes. Design values might include guiding assumptions about the nature of the design process, expected effect on users, or the interplay between the two.

In this classification, we predominantly consider the types of values advocated for designers in design processes. For instance, in user-centered design, the core values are usability and usefulness in the design process (Muller & Czerwinski, 1999). Participatory design (Muller & Kuhn, 1993) is similar to user-centered design and values the users not only for their eventual role in using the designed technology but as part of the design process from the very beginning. Participatory design is a particularly good example of how values yield a method; the idea springs from industrialization in European factories and labor politics around mechanization and automation and derives from values of democracy and socialism (Ehn, 1989). End-users of the design (such as factory workers and corporate management) are involved in the process from the beginning. Decades of work have shown how these values can be used to create a design process that involves users as constituents of and co-constructors of the design process. The social impact of participatory design is significant; often, participatory design produces mediation and communication among disparate groups as much as it makes particular

designed artifacts. If reduced simply to a set of stages or steps, it loses its effectiveness; the designer must manifest the value consistently for the approach to pay off.

A much newer value-oriented design method, learner-centered design, suggests that learning is the most important part of the design process and that human growth and change must be the most valued design goal (Jackson, Stratford, Krajcik, & Soloway, 1996; Soloway, Guzdial, & Hay, 1994). This design value is based on the assumption that people are always learning and changing and that sometimes the role of a design is not to help someone accomplish a particular task but rather to help the person grow as they adapt to the tool. Learner-centered design also explicitly notes that the user's goals may differ from the educator's goals. By recognizing this orientation as a design value, we can compare and contrast it with other design values, such as found in participatory design.

Another design method in this category is value-sensitive design (Friedman & Kahn 2000), which explicitly disregards what the designer believes to be important in favor of the values held by the target user community or society at large. Value-sensitive design is derived from ethical first principles, that is, the basic values held by society (such as self-determination or democracy) rather than instrumental values such as efficiency or expediency.

Roles

One way of structuring design is to specify the "who" rather than the "how" of designing. In some cases, the root of success or failure in a particular design process is a particular configuration of roles or a particular division of labor. Highlighting specific roles in the design process may impact the values, stages, techniques, or principles by specifying how interaction on the design team takes place. For instance, the idea of "multifunctional teaming" in manufacturing helped ensure that the values of all constituencies in a company were represented early on in the design process, from marketing to manufacturing (Shina, 1991). Another design method structured around roles is the idea of "thinking hats" (de Bono, 1999) in which participants explicitly choose different roles associated with colored "hats." These roles are not permanent but may be taken off and put on again like a hat. By making certain roles explicit and ensuring that they are filled at the right times in the design process, the method helps structure how people go about solving the design problems.

Roles are a good example of how different elements of the classification manifest themselves in different methods. For instance, in the case of participatory design, the method is relatively silent on stages, but because of the values that drive it, it has strong implications for the roles of people

on the design team, namely that users or user representatives fulfill a central role in the process, not only as informants but as co-designers. In contrast, learner-centered design, while espousing a certain value set, remains relatively silent toward the issue of roles; who is to do what is not so important.

Principles

What is meant by "design principles" is nearly as wide-ranging as what is meant by design methods. However, the most common use of the phrase is to describe rules of thumb or admonitions. These principles in some way embody not only a value for the design process but suggest a way to apply that value to a particular problem. For instance, *Apple Human Interface Guidelines* (Apple Computer, 1987), a groundbreaking work on designing a unified "look and feel" for computer applications, is essentially a collection of design principles. The difficulty of defining a design principle is in part due to the difficulty of pinning down a particular rule that applies to a variety of situations. In the most general case, design principles can become design "aphorisms" (Cooper & Reimann, 2003), or koan-like inspirations for design that are open to interpretation. For instance, "don't violate user expectations" is a highly generalized principle. On the other end of the spectrum, design principles can tend toward specifications for a particular system or set of systems that essentially determine what kinds of designs are allowable. For instance, "When an item from a pull-down menu is selected, it should blink three times before the menu disappears." A principle typically describes a rule to be followed in order to achieve a certain effect or match a certain situation. In this sense, design principles can be viewed as testable (albeit abstract) propositions (Underwood et al., 2005). One can trace principles all the way back to Vitruvius and rules such as using certain proportions to achieve a sense of visual delight in the design of buildings.

Demonstrating a more education-specific approach to design principles, Koedinger (1998) proposes design principles for learning technology in mathematics, including emphasizing mathematics as a modeling language. Likewise, Linn (1995) proposes four design principles, including deep (not broad) goals for learning, social supports for learning, making thinking visible, and helping learners act autonomously.

Both Pea and Gomez (1992) and Scardamalia and Bereiter (1991) offer design principles that emphasize the design of technologies for learning that are collaborative, embedded in a context of inquiry that transcends traditional school practices, and takes advantage of communications technology to allow communities of practice to evolve and students to participate more actively.

Generally, design principles do not crop up in a specific phase of design but serve to help constrain the problem throughout the process. Other times, they constrain the process itself. (Design principles may be most visible and explicit, however, in any evaluative phases of the design process.) For instance, Borenstein (1991, p. 115) provides examples of design principles related to implementation processes, including the advice to "proudly cut corners" in particular ways as part of the natural prototyping process. Gould and Lewis (1985) likewise lay out some general design principles that include iterative design, empirical measurement, and early involvement with users.

The production of principles varies widely. On the formal end, we have principles embodied in standards such as the International Standards Organization (ISO) standards for user interface design (the recently revised ISO 9241). These principles are negotiated and used almost like legislation, whereas on the other end of the spectrum there are participatory ways to make and share principles less formally. The educational Design Principles Database (DPD) is a good example of how a distributed community can propose principles and examples for sharing where individual designers can refine them or refuse to use them (Kali, 2006). At least as much importance is placed on examples as on clearly stating the principles themselves, and the designers themselves can add commentary freely, wikipedia-style.

Patterns

Design patterns are a type of design knowledge similar to cases or stories. A design pattern is a template solution to a common problem. Alexander et al. (1977) first conceived of design patterns in architecture; in identifying templates for solving common architectural problems, they created an "alphabet" of sorts, a set of solutions that could be combined in various ways to solve a particular problem in a "pattern language." In engineering, the cantilever is a common pattern that solves certain problems, and though the engineer will need to work out the details, certain parts of the solution are specified by the pattern. In software engineering, design patterns might include algorithms or skeletons for computer instructions; again, the expert needs only to apply the template or pattern to solve the problem without working everything out from scratch. Psychological research supports the idea that experts use patterns to simplify the solution of design problems (Schank, Linn, & Clancy, 1992; Soloway, 1985). The difference between design patterns and design cases[3] is that design cases relate one or more specific examples, whereas a pattern is the abstracted solution to the more general problem. Cases usually contain design rationales, while patterns may or may not.

Several efforts have attempted to cull useful patterns for educational software design. The E-Len project, funded by the European Union, brought together eLearning designers from across the continent to document common patterns used in online courses at universities and colleges.[4] Similarly, Fincher, Petre, & Clark (2001) used a review of computer science curricula in the United Kingdom to assemble a library of pedagogical patterns for teaching computer science; some of these patterns involved technologies as part of the pattern, but the majority were patterns of activity (rather than patterns of artifacts) that helped solve problems in teaching computer science to postsecondary students.

One challenge to the dissemination of patterns is that there are not many traditional publication venues that invite writing them up; by the time something is a well-accepted pattern, it has usually already come into widespread use via sharing of examples. For instance, the pattern of "assign readings, then assign participation in an online threaded discussion (usually with a certain number of posts of certain sorts required by a deadline)" is a common pattern in distance education. Specific cases of this are easy to locate (both in practice and in literature) whereas the abstracted pattern has become so obvious that it is never written up as such.

Techniques

Techniques are a catch-all category that include tips or tricks used by designers that are usually disconnected from more general principles, patterns, roles, stages, or values. Design techniques are often specific activities that might happen in a design process. Unlike those who describe stages of design, the proponent of a design technique may or may not care whether this activity they propose takes place at a certain point in time in the design process. Designers may plan for the techniques, or they may opportunistically pull them from their repertoire when they reach a particular sort of design situation. For instance, brainstorming might be a design technique, but this technique might find applicability at the beginning, in the middle, or at the end of any particular design process. Another design technique might be "keep a photograph of a sample user on the wall of your office while designing" or "when stuck for ten minutes or longer, take a walk." Often, techniques are useful for escaping a particular kind of pitfall or for addressing a particular kind of design challenge. Design techniques tend to be highly individualized, depending on the context, style and personality of the particular designer. The *Oxford American Dictionary* defines a technique as "a way of carrying out a particular task" or "a skillful or efficient way of doing or achieving something." Individuals pick up techniques either through invention or by adopting/adapting them from others.[5]

Design Psychology

Design psychology concerns the cognitive tricks and traps of which individual designers, or groups of designers, should be aware. This knowledge can help the designer anticipate and head off negative issues or reinforce positive ones. An example of a cognitive trick is mental simulation, which experts use to anticipate the performance of and consequences of competing routes to a solution (Adelson, 1989). Designers who use this trick can downplay details and accurately guess what things can be resolved later in the design process (Boulanger & Smith, 2001). Another trick is recognizing and externalizing one's motives for design decisions or "moves" and how these moves benefit particular stakeholders. An example of a cognitive trap is the phenomenon of "groupthink"—how individuals can converge on suboptimal solutions through peer pressure and from a desire for harmony.[6] An important part of design psychology is accepting that design rarely goes according to plan and being comfortable with the fact that there is no "ideal" process to follow.

Implications for the Design Instructor

Given these seven categories of method for the design of learning technologies, how does the instructor convey the important ideas? We propose the following six guidelines for facilitating design knowledge in learners.

First, instructors should find a way to offer students a first-hand, authentic design experience during part or all of the course. Design research clearly points to learning-by-doing as the most typical way to convey the craft of design. Most commonly, this takes the form of a "project course" wherein students, typically in teams, are required to envision learning technology solutions, often for an educational "client." In Chapter 8, we provide detailed suggestions on how to situate a project course in a "design studio," which allows experts and novices to share their understandings of problems, and focus on process as well as product.

Second, design instructors need to explain what a professional designer has automatized. People learn to accomplish things quickly and easily by incorporating ways of thinking into automatic (e.g., "chunked" or "compiled") ways of perceiving the world and acting upon it (Anderson, 1987). Operating an automobile is trivial for those who have internalized the rules and techniques of driving, though it may be very difficult for a

novice. Conversely the expert may not have an easy time explaining how to shift a manual transmission. As Cox states,

> When I've found myself in that situation, when a student doesn't understand what I can actually verbalize, then I say, "Let's work it together, and when you want me to explain some bit of my approach piecemeal, stop me," and it's easier than trying to reproduce the whole reasoning in depth.

Teachers should be experts at explaining and externalizing the automatized ways of knowing and doing such that novices can understand.

Third, an instructor needs to familiarize the design class with a set of canonical examples of design solutions, which provide a shared vocabulary or "common ground" (Clark, 1992) between the novice and expert designer. Such examples provide a way to anchor discussions about similarities, differences, and possibilities in various designs. In the typical architecture studio, this common ground might begin with a carousel of slides of existing buildings; in a software design studio for learning technology, it might include references to exemplary educational software produced professionally or in class. The Gorp Web site (see Chapter 13) and projects featured at the end of this volume illustrate two mechanisms for collecting and distributing these kinds of examples.

A fourth way whereby instructors can facilitate design knowledge is by helping students to recognize and apply patterns. The learner must develop a way to decompose problems and to recognize important triggers for particular types of solutions. As described earlier, patterns involve known generalizable solutions to common design problems, so a design instructor can use pattern libraries (e.g., Schank et al., 1992) to help students develop ways of noticing certain classes of problems and their common solutions. Note that these patterns should be more abstract than the examples that preceded them. Typically, a student would really begin to understand these by noticing the recurrence of certain solutions in the context of particular types of problems or by incorrectly trying to apply those solutions in other cases and learning to distinguish when each pattern is relevant.

Fifth, instructors can help students predict the outcomes of designs, and as they compare their predictions with actual outcomes, to refine their mental models over time to be more accurate. Though design inherently involves responding to unanticipated outcomes, the ability to make educated guesses about possible futures can be honed. For instance, students can make predictions before systematic user testing or could write up detailed scenarios predicting how designs would actually be used. Postmortems can help students compare their predictions with actual outcomes.

Sixth and finally, instructors need to support student reflection in the design process. This reflection process can be used to monitor progress, to internalize experiences from prior design problems, and to help students develop their own mental techniques and proclivities for professional, "designerly" thought and action (Argyris & Schön, 1991). The designer needs to be able to articulate his or her motives during decision making, and reflection is one way to get better at this. Reflection by dissociating oneself upon completion is an appropriate way to sift for lessons learned. In the context of a design class, an instructor can encourage reflection by building pauses into a project-based course syllabus and by scheduling final project deliverables several days before the end of the term. Asking students to keep design journals is another technique.

Summary

Because design knowledge differs qualitatively from knowledge in other disciplines, it has to be taught differently from other disciplines. The creation of learning technology requires not only scientific knowledge but craft knowledge and *phronesis* (the setting of values)—a political rather than scientific act. In contrast to other domains such as engineering, physics, or even social science, the bulk of design pedagogy may be viewed as coming from two core themes: seeing examples (good and bad), and learning "design methods" both propositionally and as skills. As design methods are so varied, we have offered here a way of grouping related methods into the basic categories of stages, values, roles, principles, patterns, techniques, and design psychology. In addition, we have provided a list of ways in which teachers can try to make explicit the tacit knowledge behind designing, and some of the ways that this knowledge can be fostered besides simple "telling."

As we as instructors aim to make tacit design knowledge explicit, we should bear in mind that ultimately, experience is probably the best teacher of design. Practice and iteration, first while scaffolded (Collins et al., 1991), then while conscious of purpose, and later once automatized, are critical steps along the path to professional practice. Rules, facts, or procedures that we share with our students are only the beginning of the journey, as it is only through experience that designers learn to respond productively to the inevitable contextual wrinkles of design processes. Thus, our primary motive as teachers of design should be to give our students as much experience as possible and as many *kinds* of experiences as possible, including vicarious ones via storytelling and observation. By embedding design methods and tacit design knowledge in experience, we

help our students join the community of designers initiating, developing, and honing their designerly habits. Design knowledge may not be easy to pin down, but it does exist, and it can be passed on if we respect how it is different from other ways of knowing.

Notes

1 Unfortunately, the phrase "design studies" is sometimes used to describe research that involves a significant amount of designing. This meaning is better captured by the term *design-based research* (Design-Based Research Collective, 2003; Hoadley, 2002). Here we use the term *design studies* to describe empirical social science research on designers, as in the journal *Design Studies,* which publishes exactly this sort of research. Compare this also with the term *design science*, which can include both empirical research in service of design and studies of designers or designing (Hevner, March, Park, & Ram, 2004).

2 This classification was originally developed by Hoadley in 1998 and has undergone refinement by looking at the literature, looking at what experts do, and looking at what is taught and learned (Hoadley & Cox, 2005; Hoadley & Kim, 2003).

3 This is another instance in which terminology is used inconsistently. Some designers talk of "essential use cases," which are basically patterns (Stone, Jarrett, Woodroffe, & Minocha, 2005).

4 See http://www2.tisip.no/E-LEN/ for their Web site.

5 One place where design techniques for software are shared is on blogs, such as the widely read "37signals signal vs. noise" blog (http://www.37signals.com/svn) and "Joel on Software" by Joel Spolsky (http://www.joelonsoftware.com).

6 We should note that studies have shown that an awareness of some negative phenomena such as groupthink does necessarily prevent them (Janis, 1988).

Bibliography

Adelson, B. (1989). Modeling software design within a problem-space architecture. In S. Newsome, W. Spillers, & S. Finger (Eds.), *Design theory '88*. New York: Springer-Verlag.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A pattern language: Towns, buildings, construction*. New York: Oxford University Press.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192–210.

Apple Computer. (1987). *Apple human interface guidelines: The Apple Desktop Interface*. New York: Addison-Wesley.

Argyris, C., & Schön, D. A. (1991). *Theory in practice: Increasing professional effectiveness* [1st Classic Paperback ed.]. San Francisco, CA: Jossey-Bass Publishers.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer, 21*(5), 61–72.

Borenstein, N. S. (1991). *Programming as if people mattered: Friendly programs, software engineering, and other noble delusions*. Princeton, NJ: Princeton University Press.

Boulanger, S., & Smith, I. (2001). Multi-strategy workplace navigation for design education. *Design Studies, 22*(2), 111–140.

Clark, H. H. (1992). *Arenas of language use*. Chicago, IL: University of Chicago Press.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator* (Winter), 6–11, 38–46.

Cooper, A., & Reimann, R. M. (2003). *About face 2.0: The essentials of interaction design*. New York: Wiley.

de Bono, E. (1999). *Six thinking hats*. New York: Back Bay Books.

Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher, 32*(1), 5–8, 35–37.

Dick, W., Carey, L., & Carey, J. O. (2001). *The systematic design of instruction* (5th ed.). New York: Longman.

Ehn, P. (1989). *Work-oriented design of computer artifacts*. Stockholm: Arbetslivscentrum.

Fincher, S., Petre, M., & Clark, M. (2001). *Computer science project work: principles and pragmatics*. New York: Springer.

Flyvbjerg, B. (2001). *Making social science matter: Why social inquiry fails and how it can succeed again* [S. Sampson, Trans.]. New York: Cambridge University Press.

Friedman, B., & Kahn, P. H. Jr. (2000). A value-sensitive design approach to augmented reality. In W. E. Mackay (Ed.), *DARE 2000: Design of augmented reality environments* (pp. 163–164). Cambridge, MA: MIT Press.

Goel, V. (1995). *Sketches of thought*. Cambridge, MA: MIT Press.

Gould, J. D., & Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28*(3), 300–311.

Hartson, H. R., & Hix, D. (1989). Human-computer interface development: Concepts and systems for its management. *ACM Computing Surveys, 21*(1), 5–92.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75–105.

Hoadley, C. (2002). Creating context: Design-based research in creating and understanding CSCL. In G. Stahl (Ed.), *Computer support for collaborative learning 2002* (pp. 453–462). Mahwah, NJ: Lawrence Erlbaum Associates.

Hoadley, C., & Cox, C. D. (2005, June 27). *Educating reflective learner centered designers*. Paper presented at the Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2005, Montréal, Canada.

Hoadley, C., & Kim, D. E. (2003). Learning, design, and technology: Creation of a design studio for educational innovation. In A. Palma dos Reis & P. Isaís (Eds.), *Proceedings of the IADIS International Conference e-Society 2003* (pp. 510–519). Lisbon, Portugal: International Association for the Development of the Information Society.

Hutchins, E. (1995). *Cognition in the wild*. Cambridge, MA: MIT Press.

Jackson, S. L., Stratford, S. J., Krajcik, J., & Soloway, E. (1996). A learner-centered tool for students building models. *Communications of the ACM, 39*(4), 48–49.

Janis, I. L. (1988). Groupthink. In K. Ralph (Ed.), *Managing professionals in innovative organizations: A collection of readings*. Cambridge, MA: Ballinger Publishing/Harper & Row.

Kali, Y. (2006). Collaborative knowledge building using a design principles database. *International Journal of Computer Supported Collaborative Learning, 1*(2), 187–201.

Kelley, T., & Littman, J. (2001). *The art of innovation: Lessons in creativity from IDEO, America's leading design firm* (1st ed.). New York: Currency/Doubleday.

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science, 18*(4), 513–549.

Koedinger, K. R. (1998). Intelligent cognitive tutors as modelling tool and instructional model: Position paper for the NCTM Standards 2000 Technology Conference. Retrieved September 6, 2000, from http://www.carnegielearning.com/nctm2000.html.

Linn, M. C. (1995). Designing computer learning environments for engineering and computer science: The scaffolded knowledge integration framework. *Journal of Science Education and Technology, 4*(2), 103–126.

Maeda, J. (2006). *The laws of simplicity*. Cambridge, MA: MIT Press.

Moran, T., & Carroll, J. (1996). Overview of design rationale. In T. Moran & J. Carroll (Eds.), *Design rationale: Concepts, techniques, and use*. Mahwah, NJ: Erlbaum.

Muller, M. J., & Czerwinski, M. (1999). Organizing usability work to fit the full product range. *Communications of the ACM, 42*(5), 87–90.

Muller, M. J., & Kuhn, S. (1993). Participatory design. *Communications of the ACM, 36*(1), 24–28.

Pea, R., & Gomez, L. (1992). Distributed multimedia learning environments: Why and how? *Interactive Learning Environments*, 2, 73–109.

Rittel, H., & Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Sciences, 4*, 155–169.

Scardamalia, M., & Bereiter, C. (1991). Higher levels of agency for children in knowledge building: A challenge for the design of new knowledge media. *Journal of the Learning Sciences, 1*(1), 37–68.

Schank, P., Linn, M. C., & Clancy, M. J. (1992). Supporting Pascal programming with an on-line template library and case studies. *International Journal of Man-Machine Studies, 38*(6), 1031–1048.

Schön, D. A. (1992). Designing as reflective conversation with the materials of a design situation. *Research in Engineering Design, 3*(3), 131–148.

Shina, S. G. (1991). Concurrent engineering: New rules for world-class companies. *IEEE Spectrum, 28*(7), 22–26.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.

Soloway, E. (1985). From problems to programs via plans: The content and structure of knowledge for introductory LISP programming. *Journal of Educational Computing Research, 1*(2), 157–172.

Soloway, E., Guzdial, M., & Hay, K. E. (1994). Learner-centered design: The challenge for HCI in the 21st century. *Interactions, 1*(2), 36–41.

Stone, D. L., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). *User interface design and evaluation*. Boston, MA: Morgan Kaufmann.

Underwood, J., Hoadley, C., Stohl, H., Hollebrands, K., DiGiano, C., & Renninger, K. A. (2005). IDEA: Identifying design principles in educational applets. *Educational Technology Research and Development, 53*(2), 99–112.

Whitford, F. (1984). *Bauhaus*. London: Thames and Hudson.