Knowing How To Design Systems

Jeffrey V. Nickerson
Stevens Institute of Technology
jnickerson@stevens.edu

1. Introduction

Teaching design is a baffling activity. Sometimes students respond quickly to training, and sometimes they appear to learn nothing. In the context of this workshop on *Multi-Disciplinary Systems Design Knowledge*, teaching design can be thought of as an attempt to impart knowledge. With this viewpoint, the classroom becomes a laboratory in which these attempts to impart knowledge, and the students' attempts to acquire knowledge, can be observed. I think that classroom research is one way of getting at design knowledge, and in this paper I suggest some of the questions that might be answered through this approach.

Pea has pointed out that learning is a sense making activity [1]; students are often confused, and they unconfuse themselves, sometimes in their conversations with instructors and with each other. Therefore, understanding how design knowledge is built may call for understanding a set of connected processes. In classroom settings, we would like to understand how the students absorb what the teacher says. And we would like to understand how the teacher evaluates what the student does. In addition, we want to understand how the students, talking to each other, make sense of the experience. There are other processes; in classes with teaching assistants, the assistant plays an integrative role.

In industrial settings, systems design is usually taught through an apprenticeship or mentoring process. This process may be less formal, but includes some of the same interpretation and discussion processes.

2. Research Questions from Teaching

I have created a course called *Integrating Information Systems Technologies*. The title comes from the ACM Curriculum of 2000 for a Masters in Information Systems, [2]. More details on the course, and proper acknowledgement to all those involved, can be found on the course website [3].

The techniques used in the class have been inspired by my previous education in design; I attended UC Berkeley's College of Environmental Design as an undergraduate and also Rhode Island School of Design as a graduate prior to studying computer science. In both schools, and in all design schools to my knowledge, design is taught through a series of *design crits*. Student work, sometimes as individuals, and sometimes as teams, toward a deadline. The students produce posters, which they hang on a wall, and the posters are critiqued by professors, visitors, and fellow students.

This technique works well in the design disciplines. Students are motivated to put in effort by the public nature of the crit; for not only the instructor but their peers will see the work. Also, peers like to see what their peers are doing, and pick up techniques from each other. The discussions let the instructor help students understand the way designs are judged, and the ways designs can be improved. The presentation and discussion is similar to the presentations and discussions students will have later in their career as they present designs to clients.

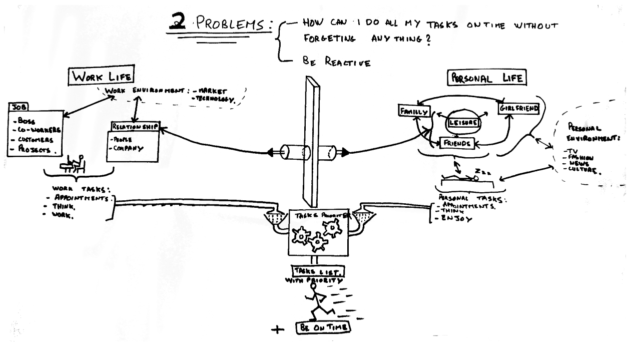

Figure 1. Part of a student poster showing a design for a personal information system

To my knowledge, this technique has not been used before on system design. However, there are several reasons to think that it might work. For systems design is often practiced as a visual activity. Designers often learn by working on white boards with more expert designers.

In addition, the problems to be solved in systems design are similar to the problems solved in environmental design. The clashing priorities of clients, the tradeoffs of cost and function, the need to realize a concept with engineering, are all similar.

However, there is a significant difference. Architecture calls for solutions to be built in three dimensional Euclidean space, whereas much of systems design is mapped out on a topological space of components connected through electric signals, in which physical placement is often less important than logical connectivity. This difference leads to the following research question:

Does design knowledge from architecture apply to systems design?

The question is significant, for our knowledge of designing the environment has been increasing over thousands of years, and is greater than our knowledge of designing systems. On the one hand, we can argue that architectural design knowledge applies to systems, as the cognitive process of reflecting on what one has designed and then modifying it is the same regardless of the domain. This stance is supported by the analysis of professional practice [4]. On the other hand, we know that software projects are much less successfully managed than building projects. There is a complexity, and invisibility, to software that is not present in traditional building. This suggests that perhaps software might call for a different design process than architecture. These two differing observations present a paradox for future research to resolve.

Some of the ideas from architecture may apply, but just as architects need to understand structural engineering, systems designers need to understand software engineering. There are underlying technologies which serve to integrate components, such as publish and subscribe mechanisms, as well as more formal analyses of processes that come from coordination science [5]. Scenario based approaches offer a systematic way of defining problems and solutions [6]. Software modeling techniques such as UML describes a set of diagrams, and systems engineering provides some heuristics [7].

Yet the reading of this material may not necessarily produce a student knowing how to design systems. There is a strong experiential aspect to design, and therefore design courses usually, and rightly, emphasize practice over reading. This leads to the second question:

How do we know that someone knows how to design a system?

As instructors a first step is to look at the students' representations. In this class we can look at posters which students bring in to the classes, as in figure 1. We also can look at sketches performed in class as part of short exercises, as in figure 2.

In the domain of architecture, Suwa and Tversky have shown that sketches are indicators of overall design capability [8]. It may be possible to apply their ideas to systems design. For example, it appears that the student with a drawing on the right side of figure 2 has a better grasp on designing systems than the student with a drawing on the left. The student on the right has connected things, and has recognized the need to store certain information.

As students have different cognitive styles, we also ask students to generate textual scenarios. In our diagnostics, we have found that students have trouble generating specific scenarios. All the students can say something along the lines of "the system is too slow", but many students have a problem writing a scenario in detail, such as "when Jane arrived and booted up her machine, it took twenty minutes before she could run a spreadsheet".

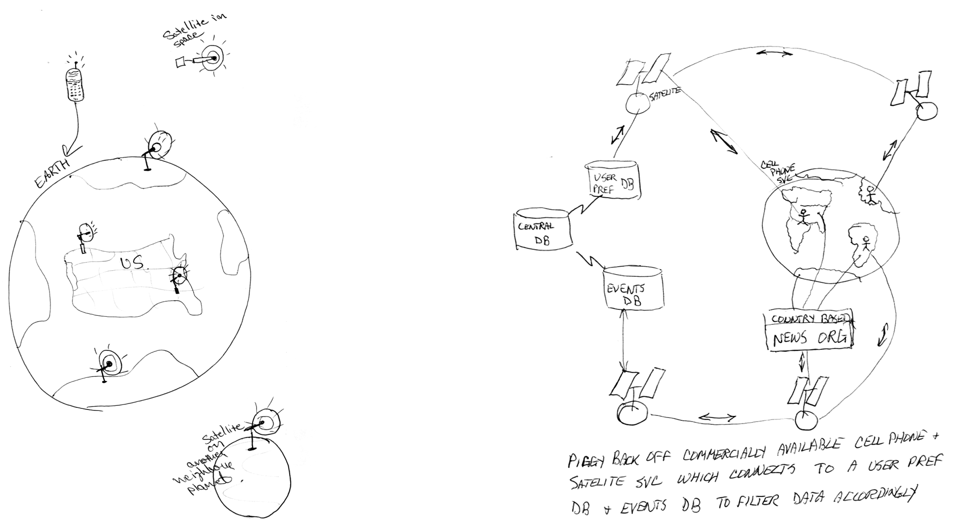
Satellite in space
EARTH
U.S.
Satellite on another neighboring planet
SATELITE
USER PREF DB
CENTRAL DB
EVENTS DB
CELL PHONE SVC
COUNTRY BASED NEWS ORG
PIGGY BACK OFF COMMERCIALLY AVAILABLE CELL PHONE + SATELITE SVC WHICH CONNECTS TO A USER PREF DB + EVENTS DB TO FILTER DATA ACCORDINGLY

Figure 2. Two students with different designs for the same problem. The students were asked to design a notification system for the international staff of a large news organization.

Some actively resist creating specific scenarios. They protest that they have been previously trained to generalize problems, and the idea of making the problem very specific seems wrong to them.

Sense making approaches to education argue that previous concepts have to be cleared away before new ones are inserted. Clearing this particular concept away as has proved difficult.

Experienced system designers often work bottom up; this bottom up aesthetic, with appreciation for the particulars of a problem, is part of hacker culture [9]. Yet in management schools where information systems design courses are offered students are often taught to generalize, and to work top-down. To teach the appreciation of specifics is to preach a design aesthetic that is sometimes counter to what has been learned before. This leads to following question:

> *Do we effectively teach an aesthetic when we teach design?*

If we do, then we are introducing students to what Monteiro calls design culture [10], and our job is a large one. One way of introducing culture is to show many examples, which leads to the following question:

> *Does sharing representations of designs help students design?*

When we look at many representations, we see patterns. In architecture, the perception of patterns was made popular by Christopher Alexander [11], and these ideas where taken up in the software community [12]. In coordination science, we see a similar approach to cataloguing business processes [13]. In architecture training, students are expected to look at lots of drawings – of their classmates, and of accomplished architects. Does looking at this work help? One can argue it should. One can also argue that seeing design and being able to produce it are very different skills.

In getting students to produce, architects focus on process.

> *Does a focus on generating and refining variations produce better systems designers?*

It seems logical that a focus on the process of design may yield results. However, there are two contradictory sets of evidence. One is that the ability to generate variation is important in design. The other is that experts in any field don't really generate alternatives – instead, they recognize situations [14].

3. A Position

Design knowledge resides in the minds of designers, who are part of a larger design culture. One way of gaining insight into this knowledge in the domain of systems design is by observing how students learn. By paying attention to the way students' representations of problems change over time, we might gain insight into how designers know.

References

[1] R. D. Pea, "Learning scientific concepts through material and social activities: Conversational analysis meets conceptual change," *Educational Psychologist*, vol. 28, no. 3, pp. 265-277, 1993.

[2] J. T. Gorgone and P. Gray, "Model Curriculum and Guidelines for Graduate Degree Programs in Information Systems: Report of the Joint ACM/AIS task Force on Graduate IS Curriculum," 2000.

[3] J. V. Nickerson and S. Desai, "Integrating Information Systems Course Web Site," http://www.stevens.edu/integration 2004.

[4] D. A. Schèon, *The reflective practitioner : how professionals think in action*. New York: Basic Books, 1983.

[5] K. Crowston, "A Taxonomy of Organizational Dependencies and Coordination Mechanisms," http://ccs.mit.edu/papers/CCSWP174.html 1994.

[6] J. M. Carroll, *Scenario-based design : envisioning work and technology in system development*. New York: Wiley, 1995.

[7] M. Maier and E. Rechtin, *The art of systems architecting*, 2nd ed. Boca Raton: CRC Press, 2000.

[8] M. Suwa and B. Tversky, " What do architects and students perceive in their design sketches? A protocol analysis," *Design Studies*, vol. 18, no. 4, pp. 385-403, 1997.

[9] D. Crocker, "Making Standards the IETF Way," *StandardView*, vol. 1, no. 1, 1993.

[10] E. Monteiro, "Scaling information infrastructure: the case of the next generation IP in Internet.," *The Information Society*, vol. 14, no. 3, pp. 229-245, 1998.

[11] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language : towns, buildings, construction*. New York: Oxford University Press, 1977.

[12] E. Gamma, *Design patterns : elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley, 1995.

[13] T. W. Malone, K. Crowston, and G. A. Herman, *Organizing business knowledge : the MIT process handbook*. Cambridge, Mass.: MIT Press, 2003.

[14] C. E. Zsambok and G. A. Klein, *Naturalistic decision making*. Mahwah, N.J.: L. Erlbaum Associates, 1997.