CHAPTER 13

The Blind Men and the Elephant, or the Race of the Hobbyhorses

Mark D. Gross

Carnegie Mellon University

CONTENTS

13.1 INTRODUCTION

Modeled on the Delft Protocol Workshop of 1994, the Studying Professional Software Design (SPSD) Workshop at Irvine asked us each to examine the SPSD design protocols using a specific method of analysis or perspective. The chapters in this book that result from the workshop remind me of the tale of the blind men and the elephant—one feels the leg and says an elephant is like a pillar; another feels the trunk and says an elephant is like a rope, and so on. Some authors showcase the felicities of their method or their theoretical framework. Others argue, on the basis of their analysis, for their design support tool. In short, like all the authors in this book, the authors in this section ride their hobbyhorses.

And that is all right. That is what we were asked to do. Our project, recall, has two purposes: on the one hand, to understand designing; on the other hand, to understand what the various methods and tools of analysis have to offer design research. In that spirit, we begin with an overview of the questions, methods, and findings of each of the four chapters in this section, followed by a few reflections on what we might (or might not) learn from them about design.

The chapter by Ball and colleagues (Chapter 14, this volume) analyzes the relationship between complexity and uncertainty of design requirements and the strategies that designers employ to work the problem. Previous design research work has generally found that, in contrast to beginning designers, experts tend to work breadth first. Yet, sometimes they do not. Which do they do, when, and why?

Ball et al. (Chapter 14, this volume) posit that, faced with requirements that they find complex or uncertain, designers choose to explore depth first. Ball et al. state, "… the preferred strategy of expert designers is a top-down, breadth-first one, but they will switch to depth-first design to deal strategically with situations where their knowledge is stretched." They also posit that when designers find requirements to be complex or uncertain, they engage in mental simulation. They describe mental simulation as "A mental model 'run' in the context of software design typically entails the designer engaging in detailed, reflective reasoning about functional, behavioral or end-user aspects of the developing system."

To investigate these hypotheses, Ball's team analyzed the written transcripts of the three SPSD study teams, and coded them for the three things that they are interested in: requirements, simulation, and uncertainty. Key words (such as "complex," "control," and "crash") identified 22 requirements from the brief, with 3 levels of complexity for each requirement (high, intermediate, and low). They used similar key-word methods to code for uncertainty and mental simulation in the transcripts.

So when did the designers engage with which category of requirements? Early in each episode, the designers brought up all the requirements categories, supporting the proposition that designers begin working breadth first. The designers addressed high-complexity categories in the first half of each episode, but left them relatively alone in the second half. This supports the proposition that high-complexity requirements promote a depth-first investigation. Also, the teams engaged more in mental simulation when discussing high-complexity requirements than intermediate- and low-complexity ones. In short, faced with gnarly problems with uncertain requirements, designers dig deep and simulate.

Matthews's chapter (Chapter 15, this volume), "Designing Assumptions," examines the roles that assumptions play in designing, using a textual analysis. "The purpose of this particular investigation is to document a set of designers' 'languaging' practices—social actions done through talk—in the course of designing." Matthews focuses on assumptions: "the real time emergence of assumptions in design activity and the work such assumptions do."

He looks at the play between two kinds of work that the transcripts of the design conversations reveal. The first kind of work is "social/interactional" and the second kind is "design." Matthews argues that these two kinds of work are inextricably intertwined and a proper analysis must take both into account. Designers' talk cannot be abstracted or extracted from its social/interactional context. As Matthews states, "… to focus exclusively on the content—of what ideas are introduced, of what considerations are taken into account, of what problems are anticipated—is to overlook the details of how design work actually gets done."

Here, we see the different functions that assumptions serve in the discourse. Through a rigorous analysis of the conversation, not only searching for key words but also marking significant pauses and nonverbal utterances, Matthews finds the various ways that the partners in the design conversation use assumptions. In one example, he shows that what participants label as a simplifying assumption serves at once the purpose of distinguishing between the system and the world, "(gently) offering it up for criticism" and allowing designers to "inspect the consequences for the model they are building of instantiating this 'simplification.'" But in another case, "a simplifying assumption is not a label that criticises

Downloaded by [Mark Gross] at 17:27 02 November 2013

an unrealistic proposal, but a suggested strategy that enables [the designers] to 'conserve ambiguity' … so as not to be waylaid by too many real world issues and constraints."

The chapter by Ossher and colleagues (Chapter 17, this volume) on "flexible modeling" argues for a particular kind of design support tool, which they call "flexible modeling tools." They argue through a language analysis of the transcripts of these case studies that flexible modeling tools are appropriate for software design. The authors draw a distinction between modeling tools that are informal and "flexible" (such as their BITkit software) and more traditional modeling tools that are formal and strictly structured. The disadvantage of the more traditional tools is that they require users to adhere to a previously determined system of domain objects and relationships.

The authors arrived, iteratively, at an analysis of the transcripts that identified concerns and concepts spoken out loud and drawing and pointing activities at the whiteboard. (They note that their analysis is not inter-rater reliable, and should therefore be considered "suggestive.") They also created several visualizations to help them analyze the transcripts. For example, they created a time line that highlighted specific concepts and when the designers discussed them. From their analysis and its associated visualizations, the authors observe the following:

- Concerns evolve; they do not only originate from requirements documents.
- Concern concepts are revisited throughout the design process.
- Concern domains are different for teams with different backgrounds.
- Multiple representations are used.
- Representations evolve, becoming increasingly formal and detailed.

These observations hardly challenge conventional wisdom in design, but they do support the authors' argument for their BITkit tool. The authors conclude that flexible modeling tools are appropriate for (at least these three cases of) software design.

Petre's chapter (Chapter 16, this volume) focuses on the ways that each of the three software design teams uses the whiteboard, and more generally what a whiteboard affords a design team. To compare and contrast the teams' performances and to discuss the characteristics of the whiteboard as a medium or platform for designing, Petre uses the cognitive dimensions framework initially developed by Thomas Green and subsequently refined by others (Green and Petre 1996). As its name suggests, cognitive dimensions analysis looks at the use of notations and external representations ("information artifacts") in terms of several key dimensions: abstraction, closeness of mapping, consistency, diffuseness, hard mental operations, hidden dependencies, premature commitment, progressive evaluation, provisionality, role-expressiveness, secondary notation, viscosity, and visibility.

Notable about the cognitive dimensions analysis of these three teams' design episodes is how little difference it reveals across the teams. One might expect the various teams to use the whiteboard differently, but that does not appear to have happened to any significant degree. This observation lends support to the cognitive dimensions analysis: that it is accurately

Downloaded by [Mark Gross] at 17:27 02 November 2013

gauging the characteristics of the whiteboard as a representational medium. Petre's chapter also points to the social properties of this particular representation: the role that gesture plays, if not precisely as part of the whiteboard itself, then as a social activity around it. Likewise, in discussing persistence, Petre alludes to the embedding of the whiteboard in the social context of the workplace. These and similar properties of the representation, or perhaps more accurately the representational medium, are important, but they are often overlooked.

13.2 DISCUSSION

Werner Heisenberg (1958) is quoted as saying, "We have to remember that what we observe is not nature herself, but nature exposed to our method of questioning." The methods we use to examine design processes have inherent viewpoints about the nature of design, and the authors of these four chapters have strong and divergent (yet not conflicting) viewpoints about the nature of design. My colleague Susan Finger tells the story from when she was a program director at the U.S. National Science Foundation (NSF) in the early 1980s, administering an initiative then called "design theory and methodology." Every day, she says, someone would call up to pitch a great idea. The pitches always started like this: "design is X and I am an expert in X, therefore …." By the end of her tenure at NSF, she had a long and interesting list of sentences all beginning with "design is X." Reading the chapters in this section reminds me of Susan's story.

So on the one hand, we are interested in what each approach can reveal about the design process—in this case, the design processes followed by the three teams of professional software designers in our workshop. On the other hand, each method of analysis brings to bear its own biased viewpoint, and will lead us to a different view of what design really is.

In the late 1960s, during the first wave of research on design and design methods, the dominant dialectic was between Herbert Simon's view expressed in his Karl Taylor Compton Lecture "The Science of Design" and Horst Rittel's view of design later articulated in "The Reasoning of Designers." Simon (1969) saw design as a process of searching for solutions, whereas Rittel (1987) saw designing as deliberative argument and negotiation among stakeholder viewpoints. Since then the scope of the dialectic has broadened considerably, but it echoes still in research on design, and specifically in these four chapters.

For Ball et al., looking at the behavior of expert designers, there is no question: *design is search*, pure and simple. The only question is how designers proceed in that search. How do they traverse the decision tree of design? This perspective is a most clear and direct descendant of Simon's view of design as the search of a multidimensional space within constraints.

Matthews's view of *design as social negotiation* is reminiscent of Rittel's view, but he makes an important point, familiar to us all from personal experience, that an argument among people is more than logic. The subtleties of verbal, visual, and gestural language play powerful roles in how we resolve arguments. If design is, as Rittel would have it, argument, then we had better attend not only to the content of the argument but also to its encompassing social attributes.

Ossher et al. (Chapter 17, this volume) present not so much a model of designing as an argument for a style of tool to support designing. They see *design as modeling*:

Downloaded by [Mark Gross] at 17:27 02 November 2013

making a model of a something that has yet to be built—in this case, a piece of software for simulating traffic. Model making as a central activity in designing has a long and illustrious history. Maurice K. Smith, an extraordinary designer who taught architecture at MIT for four decades, was fond of saying, "design is surrogate building." He was referring to the direct correspondence between selecting and assembling elements in a design and the process of constructing the completed building. The purpose of creating a model is to examine it and test it against desiderata. It is much easier to revise a design than to reconfigure a building or a complex piece of software.

It is a truism that the modeling tools cannot but affect the models made. Consider a spectrum of tools, from highly structured at one extreme to entirely open-ended at the other. A structured tool presupposes the elements of the solution and requires the designer to assemble them in syntactically correct ways, which ensures that the resulting designs are buildable. The cost is that the designer follows a fairly narrow prescribed sequence of steps, which (unless the designer is schooled in the method) tends to work against free exploration of the design space. At the other end of the structured-unstructured tool spectrum is the blank sheet of paper or whiteboard. Whereas a structured tool ensures that the design is at least syntactically correct, you can draw anything you like on a whiteboard—and later find that it makes no sense, or cannot be built. The unstructured *tabula rasa* requires designers to do the work, in their heads, of keeping the design constraints. "Flexible modeling" falls in the middle of this spectrum, striking a balance between the invitation to madness that the blank whiteboard presents and the straitjacket of a structured tool.

In the end, so what? Each analysis here engages solidly with the specific domain of software design; yet, they all make more general arguments about the design process. None of these chapters seem likely to be proven wrong; neither do they conflict with one another. Yet, each presents a starkly different perspective on the design vignette videos we analyzed. In this, the four chapters in this section exemplify the workshop as a whole: they contain many good ideas that have yet to come together to form a unified understanding of design.

REFERENCES

Green, T.R.G. and Petre, M. (1996) Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages and Computing* (special issue on HCI in visual programming), 7 (2), 131–174.

Heisenberg, W. (1958) *Physics and Philosophy: The Revolution in Modern Science*. Lectures delivered at University of St. Andrews, Scotland, Winter 1955–1956.

Rittel, H. (1987) The reasoning of designers. In: *International Congress of Planning and Design Theory*. Boston, MA: ASME.

Simon, H. (1969) *Sciences of the Artificial*. Cambridge, MA: MIT Press.

Downloaded by [Mark Gross] at 17:27 02 November 2013

Downloaded by [Mark Gross] at 17:27 02 November 2013