Musing About Design

W. Michael McCracken*

July 16, 2004

I am starting a dialogue on my views of designing and how I came to be a software designer. The following pages are the beginning of that dialogue. If you read it and like it, let me know. If you read it and don't like it, tell me why.

1 My Interest in Design

I have had an interest in design since I was a kid. Like most kids of my era, my family didn't have the money to buy me things I wanted and thus, I was forced to improvise. I wanted a go cart, my folks couldn't afford it, so another kid and I found a discarded reel lawn mower and made it into a go cart. I had an interest in ham radio, and messed around trying to modify my equipment and put giant antennas on my house. My rig had a bit too much spurious radiation and my neighbor always complained about my radio messing up his tv. I don't remember when I decided to be an electrical engineer, but as far back as I can remember, that was what I wanted to do. Like most kids, I didn't realize electrical engineering as taught at the university was applied mathematics and had little to do with actually building stuff.

I did have an epiphany when I took my first programming class. It was a numerical methods class in Fortran and I became a changed person when I decided computers were what I wanted to do. Note, I was in College in the late 60's and computers were hardly a commodity (I started on an IBM 709, moved to a 360, and ended up on a 370 for you history buffs). I took all the computer classes I could (about 4) in Electrical Engineering. There was no computer science at my school, though the math department had some "pretend" computer science courses.

How does design fit into this? I have always had an ability to abstract problems and develop solutions to them. At the time, I didn't know that was a major element of design, I just liked solving open ended problems. When I graduated, I was employed by GE, and learned what design really meant. My boss, on my first day of work, said to me, "Welcome to GE Mike. Now that you are over that college stuff, we can teach you how to design." I thought I already knew how to design! But, I soon realized that what I had been taught was how to analyze problems and sub-problems, not design. But somehow,

I relied on my ability to "get my hands around problems", and realized that design was primarily that. My skill in abstraction and getting my hands around problems soon had me solving harder and harder problems. In industry, they call that getting promoted. Soon I had designers working for me. My area of specialization at GE was software engineering. At that time, we were trying to figure out what software was all about, and I kept telling people it is no different than hardware design. Same problems, same processes, but with worse results.

One thing always bugged me about designing. For the life of me, I couldn't figure out why many people couldn't see problems like I did, nor could they solve those problems without significant struggle. At the time, I didn't think I was smarter than those folks, and today, I don't think that is the case either. But there was something different about them, and I started thinking about their psychological make up and their skills.

As time went on, that issue kept bugging me. In 1984, an opportunity arose to leave industry after 15 years and join academia. I looked at that as a chance to maybe figure out if software design was really different than other design fields and why some people could do it and others couldn't. That's what I've been doing for the last 20 years.

I started out like many people researching software engineering. I thought tools and technology would solve all the problems. Then I thought that the problem was the process and measurement. When looking at those areas, I kept finding things that led me to question my assumptions. In particular, the assumption that people and their ability to design software weren't central to the problem. I finally decided that I was focusing on the wrong problems and what I needed to do was to learn how people think when they are designing and how they actually learn how to design.

I'm not the only one who has the opinion that cognitive skills of designers are critical to successful design. Many researchers, better and more famous than me, have looked at those problems. Jeffries, et al did a seminal study of software design in the 70's [JTPA81]. Bill Curtis and his colleagues at MCC looked at design as a people problem [CKI88, CW90, GKC87]. Elliot Soloway also looked at designing from a human view [SE84, SAE88]. I must also include the book that got me thinking about this stuff, The Psychology of Computer Programming [Wei71], and the book that kept me thinking about this stuff, The Mythical Man Month, [Bro82]. What's interesting today is that the research bubble in the cognitive aspects of software design seems to have broken. Many unanswered questions are still around, it seems people lost interest in the field. That is an overstatement as there is still research in the field, though the number of researchers and papers has diminished. A recent and very interesting book is Software Design: Cognitive Aspects, by Francoise Detienne, [Det02].

2 How Design is My Life

The Lyf so Short, the Craft So Long to Learn.
My interests in design range from designing software to architecture and deco-

rative arts. The above motto was used by Gustav Stickley in the early 1900's. Gustav Stickley is generally recognized as the most important contributor to the American Arts and Crafts Movement. He was a furniture manufacturer who embraced the ideals of the movement as espoused by the English proponents, William Morris and John Ruskin.

In Stickley's era the motto referred to the craftsmen who made decorative objects with the care and skill of bygone craftsman. Today the meaning of the motto is still relevant to designing. Design in all forms has a craft like component to it that is concerned with implementability. An engineer can convince herself that a proposed design is functionally feasible with models, but must also determine if the design can be built. An architect can either design a structure and leave it to the contractor to figure out how to build it and hope that their design is reified in the form they desired, or by directly engaging in the process of designing and constructing. The other relevant aspect of the motto is that it takes a long time to learn to design (or craft in the 1900's).

OK, so it takes a long time to learn to be a designer (in my case a software designer). Where do the other interests in design come from?

In 1972 my wife read an article in a Doctor's office about mission furniture. She told me that is what she wanted to furnish our house with. At the time, we didn't know what we liked, but we did know what we didn't like. At the time, Mediterranean and early American was the style de jour. There was something about the design of that furniture that we didn't like, but we couldn't put our finger on it. We also had an interest in antiques and were regularly attending auctions and poking in antique shops, but not finding things we liked. The so-called mission furniture had a simplicity and elegance to its form, and it was masculine in scale. My wife has never liked frilly things. The stuff was almost antique, yet I never found it in the shops in Syracuse where we lived at the time. So at her urging I took on a quest to find mission oak. I'm still looking and learning today. Thus, my interest in decorative arts.

A natural fall out of decorative arts is architecture (at least residential architecture). My interest in architecture is two fold. The design component is a focus on late 1800's and early 1900's architecture because of the relationships to the decorative arts of the period. The other interest is in construction. Like I said earlier, I love to build things, that includes houses.

I have studied or done computer science, in particular software engineering, all of my adult life. I still feel I have so much to learn and so little time to learn it in. Software engineering in many ways is a craft. I have studied the Arts and Crafts movement for almost the same period of time, and I still have so much to learn.

3 My Take on Design

Simplicity is the Essence of Elegance

I developed that motto almost 20 years ago (at least I think I developed it) and it best describes my views of designing. To me, designing is managing

complexity. Problem complexity and solution complexity. Simplicity through abstraction or through elegant designs is how I evaluate designs. What I mean is a prime evaluation criteria of alternative solutions beyond correctness, should be simplicity. Many designers argue with my comments because they misinterpret my viewpoint. I am not saying strip out functionality to make something simple, or reduce complexity by omission. What I am saying is that a great design can be relatively simple when compared to a poor design. Frank Lloyd Wright in my mind followed that view. If you think of Fallingwater for example, it is a simple solution to a very complex problem, how to build a house on top of a waterfall. There were inherent complexities that had to be managed, but they were managed to minimize the unnecessary complexity.

4 Why it's fun to learn

Be a kid, ask why

I tell people I have the best job in the world, at least for me. If you like to learn, how much better could it be than being in a job which requires you to constantly learn? The motto, another developed by me, summarizes my views on life and learning. You obviously don't have to be a kid to ask why, but asking why is what I do. I am often referred to as an empiricist, especially by one of my favorite colleagues, Colin Potts. He's right. I don't make new theories from great thoughts, I observe and try to explain. I don't know much about Aristotle or Plato, but my guess is most intellectuals lean towards Plato's rationalism. My practicality (and probably my intellectual limits) keep me from being able to abandon empiricism. As a result, I study lots of things that don't need to be studied (at least by a rational person).

I had a discussion with a friend a few years back that tickled my fancy. It regarded the finishes used by manufacturers of Arts and Crafts furniture (mission oak). Since that day three years ago, I have been reading everything I can get my hands on relative to the finishing industry at the turn of the century. I've learned some organic chemistry, about the dye industry, about the furniture manufacturing industry, and a whole bunch of other stuff, all of which is completely irrelevant to my job. Instead of reading novels for leisure, I read technical books from the turn of the century. That's a crazy way to spend an evening, but I'm pushed to understand why. Probably a different take on Plato and Aristotle. Instead of rationalism and empiricism, maybe it's rationalism and irrationalism!

5 A bit more rationalization

If it ain't fun, don't do it

That little motto has gotten me in trouble more than it should have. You see, if it ain't fun, I tend to not do it. In many cases, that is not a good decision. Somebody's got to pay the bills. Bill paying is like having a root canal. If you

put it off, your teeth or you credit worthiness will fall out. I'd rather read a book on Art Nouveau in Prague than pay a bill. I suspect many people have a similar predilection.

There is a more serious view that I like to offer to students. Don't get yourself in a job you don't love. The way to prevent that from happening is being smarter than those around you. It takes brains, common sense, and attitude to be offered something better. When I was in industry, I was always offered better jobs. I never had to seek them out.

6 Different Designers, Different Designs

7 Why is Software Design Hard to Learn and Hard to Do?

References

[Bro82] F.J. Brooks. *The Man-Month: Essays on Software Engineering*. Addison Wesley, Reading, MA, 1982.

[CKI88] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31:1268–1287, 1988.

[CW90] B. Curtis and D. Walz. The psychology of programming in the large: Team and organizational behavior. In J-M. Hoc, T.R.G. Green, R. Samurcay, and D. Gilmore, editors, *Psychology of Programming*. Academic Press, London, 1990.

[Det02] F. Detienne. *Software Design - Cognitive Aspects*. Springer, London, 2002.

[GKC87] R Guindon, H Krasner, and B Curtis. Breakdowns and processes during the early activities of software design by professionals. In G.M Olson, S Sheppard, and E Soloway, editors, *Empirical Studies of Programmers: Second Workshop*, pages 65–82, Norwood, NJ, 1987. Ablex.

[JTPA81] R. Jeffries, A.A Turner, P.G Polson, and M.E Atwood. The processes involved in designing software. In J. Anderson, editor, *Cognitive Skills and Their Acquistion*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[SAE88] E Soloway, B Adelson, and K Ehrlich. Knowledge and processes in the comprehension of computer programs. In *The Nature of Expertise*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[SE84] E Soloway and K Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, 10(5):595–609, 1984.

[Wei71] G.M.. Weinberg. *The Psychology of Compute Programming*. Van Nostrand Reinhold, New York, 1971.