An Open Letter to the Tech Community:
Humanize the Craft of Building Interactive Computer Applications

Much of my work since the late 1950s has been motivated by my belief that an understanding of how interactive computer applications work must become part of the common culture. Over the span of human history the understanding of three important technologies has migrated from the domains of priests to the domains of elementary school teachers: arithmetic, writing, and the calendar. Now software technology must join this migration. I write this paper to suggest a first step in that direction. In this step two changes must occur:

1. The conceptual model that people use to think about the internals of interactive applications must change to something more intuitive than coding.
2. The process of building applications must change to something more aligned with the way our bodies and brains have evolved.

We must find a way for people to think about and build interactive applications that will be, in a word, *humane*.

There is more to this than computer science

My search for a new approach has led me to thinking about building applications, not from what is particular to software, but from what is universal to humanity.

The epiphany came as I watched a baby grandchild sitting in his high chair trying to eat some Cheerios spread around the tray in front of him. It was a struggle–first identifying a target, then getting his hand to where the target was, then grasping it, then navigating it to his mouth while maintaining hold of it, then opening the mouth and putting it in, and finally letting it go to the right place so his built-in eating reflexes could take over.

What an achievement! I was witnessing the main product of human evolution. Most animals are born knowing how to do most of what they will need to do to survive. Not so with humans. The particularly human miracle I was watching was not the eating but the *struggle*. I was watching the beginning of an individual

human's struggle to teach himself to use his hands as tools. The most impressive part of this miracle was the *persistence*: each person is driven to practice and practice until he gets his hands and eyes to work together to do what he wants done. For some people–brain surgeons, sculptors, and painters, for example–the struggle lasts a large portion of their lives.

The human brain is programmed to build itself.[1] It starts off with little in the way of skills but contains a built-in process for building and modifying itself through experience. Just as importantly, each human individual is given the persistence to work at this building process tirelessly, literally for years.

Humanizing application building means finding a way to take advantage of what Nature has given every normal human: a brilliant hand-eye-brain system. But this isn't all; the human goes on to use this hand-eye-brain system to build things that nobody programmed him to build. Witness a child with a paint kit or on a beach building sand castles. He is using his brain, not only to control his eyes and hands but also to conceive, and then build, novel artifacts.

This is an awesome pair of assets: a sophisticated hand-eye-brain system and the capacity to use it to build inventions. This pair of assets is a large part of what characterizes our species, and *each one of us is the beneficiary of this gift*.

> Choosing to exploit this investment in our species has been my starting point for rethinking the craft of building software applications.

Hand-eye-brain tracking

The skill that my grandson was teaching himself can be called *hand-eye-brain tracking*. It involves a tight feedback loop: the brain chooses to move the hand; the eye and sensors in the hand and arm tell the brain how the hand is incorrectly placed; the brain sends corrective signals to the muscles. This communication loop is operating continually; viewing it as a process, one sees a flow that maintains a continual and accurate correspondence between the hand movement that the brain visualizes and that actual movement of the hand. It is the classical negative feedback loop employed by all kinds of control systems, from refrigerators to

[1] https://www.ted.com/speakers/michael_merzenich

guided missiles, that maintain a tight dynamic correspondence between goal and outcome.

Purposeful manipulation

There is a story of the homeowner whose gas water heater wouldn't stay running; it would start up and then immediately stop. He called the plumber who arrived, sized up the situation, got out his hammer, and delivered one clanging blow to the tank, upon which the water heater started up and remained running until the water was hot. Two days later a bill for $100 arrived from the plumber. The homeowner indignantly wrote to the plumber demanding that he justify with an itemized statement how banging on a tank can cost $100. The reply came back in the mail: "Hitting with the hammer, five dollars. Knowing where to hit with the hammer, ninety-five dollars."

Purposeful manipulation as practiced by an artisan has two parts:

- Strategy (knowing where to hit), in which the artisan visualizes a path of action going from the present condition to the desired state.
- Tactics (hitting), in which the artisan acts to move along the path.

Purposeful manipulation can involve simple skills (hammering together a doghouse) or exquisitely sophisticated skills (brain surgery). When the activity involves using the hands, hand-eye-brain tracking is always involved.

The artisan's unconscious dance

A skilled potter at her wheel is my ideal example of the artisan's unconscious dance. In this ideal the understanding of the working material and the fabrication process are so built into the mind-body of the potter that it seems that her hands alone are directing the continuing evolution of the working material from a formless lump of clay to an elegant sculpture.

If you break down the construction of a doghouse into small steps you might say that the unconscious dance applies to each step, for example, pounding in a nail. This is not what I mean; I am limiting my definition of the artisan's unconscious dance to a whole session of building a complex artifact, considering this session as

a unity from beginning to end. Artists and artisans who are self-observant sometimes speak of the experience of the unconscious dance as *being in the zone*.

Application building as purposeful manipulation

A software developer building an application employing coding techniques is employing purposeful manipulation. He knows the application's behavior he is trying to create, and he employs his understanding of software technology and his toolset to build an artifact that exhibits this behavior. The typical toolset consists of such things as text editors, compilers, and debuggers.

Application building as unconscious dance

Coding at a keyboard is an unnatural act for almost all of us. The reason is that it cannot be experienced as an unconscious dance.

> My thesis: to humanize application building, find a way to transform the application construction process into an unconscious dance. This transformation will require two inventions:
>
> 1. an intuitive conceptual model of application structure and
> 2. a correspondingly intuitive construction process.
>
> Used together, these two inventions will replace conventional programming behaviors by hand-eye-brain tracking.

I've come to understand this requirement well enough that I can state some design principles that must govern the application conceptual model and the construction process, and I have built a prototype that illustrates these principles. I am persuaded that observing these design principles will move us significantly closer to humanizing application development.

Design principles

There is only one form of the application and there is no translator. The unconscious dance requires that the artisan keep her hands on the working material, which she transforms in stages from its initial unformed state to its final fully formed state. Replace the notion of

input-process-output (a "source" form of the application going into a translator and coming out as an "executable" form of the application)

by

in-place transformation of the working material (a single always-working application being gradually transformed from its initial to its final form).

The notion of *starting* an application is not fundamental. During development the application is always running, even in its trivial initial state. (This discussion is limited to event-driven applications; this limitation is the motive behind the word *interactive* in the title.)

During development the developer role and the user role coexist simultaneously as peers. The application's user interface and a view of the application's structure sit side-by-side in front of the developer/user, and either the application's user interface or the view of the application's structure can receive the next event from the developer/user. Any event from the developer/user immediately leads to a new consistent state of both the application's structure and its user interface. If the application makes use of a source of external data such as a database, this can be present or absent at each stage of the transformation, and its presence or absence is always consistently shown in the application's user interface and in the view of the application's structure.

Until the video demonstration arrives
learn more at
http://melconway.com/hand-eye-brain.pdf