Post-Information Age Skills: why coding will set you free

Michell Zappa
Ethical Technology

Posted: Jun 22, 2012

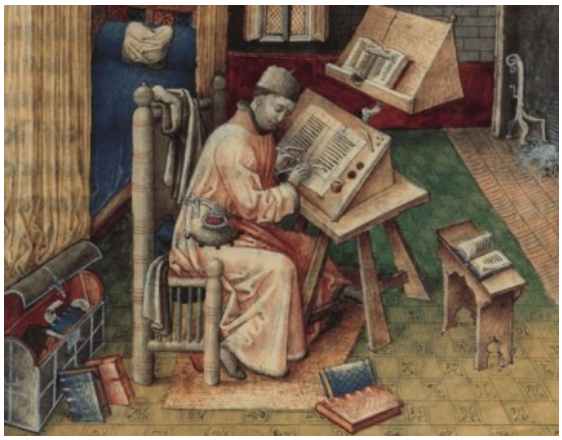
http://ieet.org/index.php/IEET/more/zappa20120622

The merits of literacy are self-evident to the point of no longer being questioned in society. The very concept of reading and writing is a tenet of social compatibility for most cultures, having embedded itself into our social fabric to the degree where even debating whether *"we should teach our kids how to read & write"* is preposterous. But one doesn't have to trace far back into our history before encountering an era where literacy was a rare skill for a very distinct minority.

Literacy

A religious technology trait, in the middle ages, writing was a skill relegated to scribes, who needed it mostly to replicate bibles. Fast-forward past Gutenberg, movable type, the industrial revolution and past the information age, and today there's no debating that a literate populace has direct economic and intellectual leverage against an illiterate one.

The issue of literacy in most western countries today is focusing on the importance of teaching programming as a fundamental life skill for future economic gain. Having already gone through a boom-bust cycle in many education systems in the 1980's and 1990's,



programming went from being a crucial computing skill to being relegated to a select group of specialists as UIs grew easier to manipulate. Today, learning C, PHP or Assembly isn't of much value for a non-developer, mostly as a function of its inherent complexity (programming is hard) and lack of broad necessity (programming is a means to an end).

I find myself in the camp of technologists seeing a fundamental shift happening around the role of programming in a post-information-age. Increasingly, the skill will transcend occupation and embed itself more deeply in aspects of daily life.

Why Programming

Programming skills are regarded as crucial to develop a thriving economy (Silicon Valley being the prime proponent of said argument), but on a more fundamental level it teaches us skills that underline the contemporary condition. For starters, code is about recursive thinking: to use logic for breaking down complex problems into smaller ones, and solving these with specific tools. Code is also about heuristic thinking: making sense of large swathes of data and information, and throwing computer power against issues our brains are not optimized at solving. And finally, code is about abstract thinking: detaching oneself from the problem at hand and analyzing it from another level. It's about defusing the world and making it seem

less magic (in response to Arthur C. Clarke's famous quip about advanced technologies being indistinguishable from magic).

"If you can code, you start to see the computer as a machine that can do anything you want, instead of just the things some app store makes available to you. That freedom is addictive. You start demanding it." Dennis Peterson, in the comments.

The Business of Code

The case for coding skills being directly tied to prospective job opportunities (as an employee) or potential capital (as a startup founder) has been made to exhaustion, and the global "Silicon Geographies" prove the point. To further the argument, I want to make the case that algorithmic skills are reaching an increasingly wider base.

Take IFTTT, a free tool allowing you to create If-Then clauses for any combination of web-enabled services, such as: "If I am tagged in a Facebook album, then save that photo to my Dropbox" or "If it is going to rain tomorrow, then send me an email first thing in the morning". Said examples are simple but of broad appeal, but the general trend is of increasing future complexity and possibilities.

Only a couple of days ago, Microsoft announced a peculiar tool called on{x}, enabling Windows Phone users with basic JavaScript knowledge to extend the IFTTT paradigm of triggers and actions to the real world. on{x} allows your phone to, for example, automatically reply to a text message from your wife containing your current location (based on prior permissions, or remind you to visit the gym if you haven't been in three days (based on your location history). Just like with IFTTT, the applications are still rudimentary, but part of a larger trend.

Mojang, the company behind smash-hit Minecraft, recently announced that their next game title would essentially be a programming language. Entitled 0x10c, the game is a spaceship simulator where the player controls a virtual 16-bit CPU using actual code. The abilities of your ship, such as its speed, cloaking capacity, maneuverability, etc are a direct function of well you manage to utilize the simulated processor using code. Evidently, the intent isn't for the title to have a broad appeal among gamers, but rather reach a very particular niche—but again, the very concept of embedding actual programming in a video game is indicative of bigger changes.

Not to mention the meteoric success of the Raspberry Pi, a British initiative for mass-producing extremely affordable, hackable computers. A tremendous hit, the $35 pocket-sized Linux machine sold its initial batch of 10.000 units in minutes, proving that the maker ethos is alive and well.

"The web is becoming the world's second language, and a vital 21st century skill — as important as reading, writing and arithmetic. It's crucial that we give people the skills they need to understand, shape and actively participate in that world, instead of just passively consuming it." Mark Surman, Executive Director, Mozilla

Algorithmic & Parametric Design

Revising the fundaments behind the importance of coding skills, it becomes clear that being educated to think along the capacities of machines is bound to have positive consequence in areas not directly related to code. Take medicine, where the requirement for bug-free engineering is paramount—or pharmaceutics, where big-data analysts are being hired in droves in order to algorithmically identify potential new drugs.

With the advent of real-time analytics, even geo-economies like real estate and global shipping are rapidly turning algorithmic, drawing upon wealths of data in order to optimize global routing and predict booming hotspots.

Some claim that 70% of trading on Wall Street is already algorithmic, what about something as subjective as fashion forecasting? Pimkie Color Forecast automatically analyses webcam feeds in Paris, Milan and Antwerp to observe the predominant new colors worn on the street in order to extrapolate potential future trends.

Not even intellectual skills like journalism are safe from the onslaught of computer mediation. Companies like Narrative Science are already mass-producing sports and finance news reports based on real-time statistics, and some predict that the vast majority of news writing will be intermediated by algorithms by the end of the next decade. Architecture, too, is rapidly moving toward so-called Parametric Design, where buildings, floor plans and façades are designed according to a delicate interplay between artist and code.

Tools for Making Tools

In short, programming is the simplest tool-making tool. It allows for rapidly improving on existing solutions as well as solving problems previously thought to be irreducible.

Technology has long become our external memory, allowing us to entirely forget rote information such as phone numbers and addresses. Increasingly, our tools are swallowing the responsibility of agency, being more capable than us at telling us **what** to do (think Siri and fast-forward). If current trends are indicative of our future, the next generation of tools will allow us to forget **how** to do things. Take the U.S. financial market with its flash crashes. Our best minds are being applied to developing systems whose actions we no longer understand and we are rapidly becoming a species developing tools whose output we do not understand.

Knowing how these systems act, and determining how they ought to behave in the future is our responsibility.

Michell Zappa is a global emerging technology strategist who publishes a project entitled *Envisioning Technology*. His focus is on explaining futurism and technological scenarios using visualizations and creativity drawn from science fiction.

COMMENTS

b. • • Jun 25, 2012

I agree here, but I think the root of “forget how to do things” is just not caring, taking the easy pragmatic choice and not thinking critically. I think that is the key thing that seems to be missing, we’re good at being critical of politics or the media, but we are not good at being critical of technology, and how it changes (often unconsciousnessly) how we think and what we consider important.

Perhaps the in depth kind of critical thinking I’m thinking of requires technological literacy (including programming) to even be possible.

IFTTT looks cool, but most “channels” are centralized corporate entities.

SHaGGGz • • Jun 29, 2012

That penultimate paragraph reminds me of that sci-fi short story set in a far future where humans have long forgotten how to operate the singleton that runs their society. Forgot what it’s called… But yeah, we’ve been on this path for centuries now, and it’s only going to deepen. Until cognitive neuro-implants, that is…

@b.: I think a large part of that lack of technological criticism is a result of not understanding said tech. However, I don’t necessarily agree that programming is needed to formulate useful criticisms. For instance, David Carr’s notion of “the shallows.”

Next entry: The Death of Alan Turing (rare documentary)

Previous entry: What Happens When We Turn the World’s Most Famous Robot Test on Ourselves?