

[iDC] The 50-Year Computer

Patrick Lichty [voyd at voyd.com](mailto:voyd@voyd.com)

Sun Sep 28 18:25:52 UTC 2008

- Previous message: [\[iDC\] Art and the Geek](#)
 - Next message: [\[iDC\] The 50-Year Computer](#)
 - Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)
-

The 50-year Computer
Manifestos for Computational Sustainability, I

I have a proposition to make - when I am ready for my first mind/body transplant in 2058, at age 95, I want to be using the same computer I am today. Upon first look, both may seem outlandish by today's standards, but I argue that the former is far less preposterous than the latter. What I want, among other things, is a computer with a fifty-year life-span. However, attaining this goal will require a fundamental rethinking of our computational culture, and will likely be at odds with manufacturing agendas put in place in the mid 20th Century. However, as we are confronted once again with issues of sustainability, paradigmatic changes are called for. The idea of a 50-year computer could be one of them. My concerns that have led me to this proposition are based on the unnecessary planned obsolescence of computers, necessity vs. desire, and the huge amount of landfill that old computers contribute.

To elucidate on what I am proposing, I would like to draw some analogies to the idea of the 50-year computer. As a quick metaphor, consider the automobile. If cars were operated under the same methods as computers, there would be three types of roads, each with its own kind of cars at this time (Windows, OSX, and Linux). And, for at least two of these road types, every five or ten years or so, all cars made before may not drive on the current type of road. For example, my father, aged 87 at the time of this writing, has a 1963 (one year after my birth) Studebaker Lark that he loves to shine up and drive every summer. It does have some drawbacks, like the fact that it had to have seat-belts installed as an option, it lacks many of the amenities found on contemporary cars, and it occasionally needs new tires (which are often hard to find) because rubber gets hard with age. It gets good gas mileage despite its eight cylinder engine, and drives quite well, if a little loose in the steering. It is certainly not a "modern" vehicle by any standard, it is still basic, solid, fun transportation despite its age. I didn't have to buy an upgrade to make it operable on current roads.

People buy computers based on desire, not based on what they need. Many of my colleagues in academia, especially in history and the humanities, have only needed some Web capability beyond the basic office suite software in the last fifteen years, as is also the case of my father. Keep in mind that I am talking about mass - market, general-purpose computing, and not more specialized media production applications, such as 3D animation, cinema editing, and the like. Yes, making your own videos and putting them on YouTube is fun, but do we NEED it from a functional perspective? I argue not. This is not to say that consumer-level media production, high-resolution games, and so on are not enjoyable, but this does not fit my proposition of an essential, appliance-grade, general computing platform

that provides elements of "essential" computing for long periods of time.

In the 1980's when I was a field engineer for Tandy Computers (now defunct as a division, but was part of the Radio Shack consumer electronics chain), there was a LAPTOP called the Model 100. It was entirely solid state, battery powered, had an extremely legible LCD screen, and was a known workhorse in the journalism field. What did it do? It had the basic office programs (word processor, database, spreadsheet) and a terminal program for online access, as well as a modem. It also had a ROM slot for additional program chips. In my research, I have seen a number of them still in use nearly thirty years after manufacture, although obvious repairs such as keyboards (one keyswitch at a time, mind you) or displays have been done. The Tandy 100 is probably the first case of my knowledge of a basic platform for long-term general computation and an initial inspiration, if not just a fine metaphor for this computational model.

General Computing?

General computation refers to the creation of a standard for basic, generic-use platforms with a set hardware criteria created to run the "road", or operating system and its set of fundamental programs. Since we are defining a standard for a computational platform designed to run a software set for fifty years (or more), we can assume that the platform is "static", or relatively unchanging. This is to say that the machine could have a basic von Neumann structure that will support a general operating system kernel. While over time some speed or miniaturization may take place, the general specification remains backward compatibility. Therefore, my 2058 Methusatech computer will be designed to run 2008 code and vice versa, which is done by adhering to platform standards. Elements which are comprised of moving parts, or parts that incur wear are built to be easily replaced. Although this is designed to drastically scale back large-scale production of electronic components, new units will ultimately be made, and a market for replacement parts will also be needed.

What Operating System?

The choice for an OS standard could be a fictional new system designed to support the platform, but the various flavors of "-ix" (Unix, Xenix, Linux) have been around for at least thirty years, and seems to be the logical candidate for the platform. The degree of stability of availability, scalability, open source standards, interoperability between platforms, and user base make it a good choice for the progenitor of the MehusalOS operating system. There are also older systems, such as CP/M, OS9 (not the Apple version), but the choice of an -ix system reflects a cultural continuity of nearly forty years.

Software-centricity

So far, we have been talking about the creation of a "static" platform and scalability of operating system, but what does that do to the production/development culture of the platform? It creates one of "software-centricity", that is, a decentralization of hardware manufacture beyond that of basic information appliance, and places all of the emphasis upon that of software development. First, stasis of hardware standards means that programmers are able to delve more deeply into the intricacies of the platform without concern for standards change. The general internal structure, libraries, etc. will remain stable or easily upgradeable on older platforms.

A great example of innovation through limitation is shown through the "demo scene", or a community of artists, musicians, and graphics hackers, who make small graphics/audiovisual demos, with the largest community in Europe. Although a full discussion of the "scene" is beyond our scope, and their

community operates on currently evolving platforms as well as "dead" ones, the 2K, 4K demo challenges exemplify a model for software-centric design. In this model, the coding team is challenged to create a fully functional demo that exists below the 2048 (2k) or 4096 (4k) byte size constraint. As hard drives what reach into the terabyte range become more commonplace (and this may seem quaint by 2020), acceptance of limitations of software size challenges the programmer to extract all possible work from every cycle. It eliminates problems of "bloatware" as evidenced by Windows Vista, but the user may still accumulate flash drives of programs and data, which is not surprising.

A Movement towards Elegance

As mentioned before, the creation of a static platform for general computing eliminates the frustrations of continual upgrading and shifts the primary focus in computer production to that of software development. This creates a culture of elegance in coding, and craftsmanship amongst coders unseen since the days of personal computing of the 1980's. One could even say that a meritocratic market could arise for "code magicians" who are masters of programming within constraints. This is evident in programs written in the latter days of platforms like the Atari 2600 game platform like BASIC Programming, Visicalc for the Atari 800 computer, and even many applications for contemporary mobile phones. Once again, the matter is that we are talking about a platform for essential computation, and not a complete replacement for the contemporary computer, although hopes might be that it may become one.

Kicking Moore out of the Temple

In 1965, Gordon Moore wrote his seminal article that would create the foundation of "Moore's Law" or that the density of transistors in integrated circuits doubles every two years. While this has led to a similar increase of the power of computation, it has also created a technoindustrial treadmill where there is a politics of fear based on the postwar design agenda of planned obsolescence which has translated into the culture of personal computation. From so many personal experiences as an educator, and noting the annual software upgrade cycle, the fear of not using the latest, greatest hardware connotes one's relegation into the "dustbin of history", as the utility/viability of computation is often linked to the desire for more (computing) power and the fear of being linked to technologies victimized by planned obsolescence. While taking our current polemic desires to make making Moore's law irrelevant, it also does not ignore the fact that what is being suggested is a genre as well as a paradigm. Pragmatically speaking for the foreseeable future, there will be desktops and larger laptops, but this paper suggests standardization of even lighter, inexpensive computing devices as is evidenced by the sub-notebooks of the late 2000's.

Use What You Need, Hardware or Software

Returning to the analogy of the old "basic four" programs, we are questioning the necessity of large media production machines and gaming desktops. To reiterate, the 50-Year Computer is not necessarily a replacement to the large desktop, but a small, set platform for personal computing. Under this paradigm, one could say that one is using "what they need" as opposed to "what they want". Those not interested in the mindset of necessity will "want" of the larger machines.

This is not to say that this new computational model is any less flexible than an iPhone. While the hardware remains static (or relatively so), the software-centric model discussed before creates more tightly coded, "concise" programs. This stresses specific utilities, as the computer is not limited by any means to the "Basic Four". In addition, there will

undoubtedly be some operating system updates, as well as firmware, given the two maxims of hardware consistency and forward/backward compatibility. Conciseness also means that each program will be more specific in their function, akin to contemporary small "apps" for portable devices. The interoperability of these apps is also part of the way around the limitations of the platform, and this is related to programmatic distribution of tasks.

Programmatic Distribution of Tasks

So, how does one deal with limitations of a static platform in regards to tasks that grow beyond the capacities of the machine, such as expanded spellchecks, etc. This is done through a set of data standards and programmatic distribution of tasks. One example is that of invoking a spellchecker if needed, even matters like image filters, as separately invokable routines. This is only a partial solution to the limitations of the system, as chaining together multiple data streams across vast archipelagoes of sub-applications will become unnecessarily cumbersome. Again, the 50 Year Computer is designed to address essential computing, is positioned against Moore's Law, and the aforementioned task-sharing is envisioned as a potential method for squeezing every last cycle's potential from the machine.

Scalability

All things being said, this writer is not a complete romantic, and understands that over the fifty years of production of the hardware between 2008 and 2058, there might be some upgrading of the platform. In such a case, the essential aspect of the platform is that of scalability and backward-compatibility. This would be done through retaining the core kernel of the operating system and consistent general hardware architecture. As envisioned, the only sacrifice that the 2008 Methusatech user would have as opposed to the user of the new 2058 model is less internal RAM (they could use external non-volatile RAM), and perhaps some speed. The hope for the system is that the operating system and hardware would be upward and downward scalable so that a relatively consistent user experience would exist between the 2008 and 2058 models of the machine.

Not Cloud or Network Computing

During the first fifteen years of the World Wide Web, on at least two occasions there have been alternate models to the desktop in the form of the Network Computer and Cloud Computing. The former, started in the 1960's and attempted as a paradigm in the 1990's, was a method in which programs would be served remotely, allowing for the majority of storage to be done through servers. The NC, was in effect, a "really smart terminal". Conversely, Cloud Computing borrows from the distributed models of SETI Online, which allows millions of users to analyze radio telescope data to determine correlations for possible intelligent life. Machines are clustered to distribute their computational power through the creation of "clouds" of processors. Although the 50-Year Computer is surely not an "NC", and it might be technically able to be used as a cloud machine, its function is envisioned as a single-person information appliance for the execution of essential personal computing.

50-Year Computer: Not a Blow to Innovation

Recently, while discussing this concept, I was challenged that this idea was against innovation, and this is patently not the case. It merely suggests a different paradigm to cultures of innovation. For example, a timeline was laid out for me outlining increases in resolution, the addition of color, increases of power mitigating increases in amenities, to which I asked to what ends these advances served. "It answered the question of what people want...", he said. Once again, we are faced between desire and utility in

the face of a Fordist production scheme that increasingly puts more toxic landfill in the 3rd World and creates an endless cycle of fear, desire, and obsolescence, and the 50 Year Computer refutes this. On the other hand, it creates a new culture of innovation in the software sector, which is a paradigmatic shift from contemporary models. These models of software and hardware production are patently unsustainable, and by allowing us to think of operating "antique" computers, we also antique the notion of planned obsolescence, even though proposing a radical new paradigm may seem ironic in this context.

Conclusion

The 50-Year Computer, or something like it, is not a "modest proposal" that merely dismisses the computational culture of Moore's Law, and technodeterminism, although it does challenge it. From my experience in computing for the past thirty years, much of what I call "essential" computing still revolves around functions served by basic office software, with the only addition being functionality for the Web. Although contemporary computation allows for wonderfully rich and engaged experiences, it's arguable as to whether many more necessary programs, including aforementioned Web technologies have been created in the past thirty years. From this, the 50-Year Computer is a statement is designed for reflection upon the grossly inefficient, wasteful, and unsustainable computational culture extant in the technological world. In proposing the 50 Year Computer, I suggest many things; a "Model T" computer that performs utilitarian functions for decades, reduction of energy consumption, and waste streams, as well as maintaining occupation of labor by redistributing it to software production. This is not to say that the 50 Year Computer will eliminate the desktop, as it is unreasonable to expect to eliminate an entire computational culture of scale merely by wishing it. The proposition of this missive is to suggest alternatives for computation that could offer greater sustainability, affordable utilitarian computing to the global masses, and cultures of software artisanship. What is clear is that as the 2010's approach, if we are to continue using computational devices, we must consider alternate methodologies that take in account the use value of computation to world masses as well as general sustainability.

-
- Previous message: [\[iDC\] Art and the Geek](#)
 - Next message: [\[iDC\] The 50-Year Computer](#)
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

[More information about the iDC mailing list](#)