Rethinking software

Suppose software could not be compartmentalized. Suppose whenever someone said “linux,” they had to say “linux is a technology.” And whenever someone was describing hardware, they had to say, “hardware is technology.” If software and hardware were not separate, linux could be a car.

Suppose plumbers could not purchase standard fittings without going through a piping company. Plumbers might face increased costs to buy plumbing equipment that only fit certain fixtures.

Software architecture, specifically, its user interface, to many users, is much like a zeitgeist. A zeitgeist“ is an invisible agent, force, or daemon dominating the characteristics of a given epoch in world history.[2] The term is usually associated with Georg W. F. Hegel, contrasting with Hegel's use of *Volksgeist* "national spirit" and *Weltgeist* "world-spirit".

To developers however, it may resemble more a volksgeist – the engine under the hood of a Volkswagen.

A century of car manufacturing has led to the “average” driver much more familiar with the basic internals of a car, with the exception of electronics, which as obscured some of the common understanding of a modern car. A century from now, it is unlikely a large percentage of computer users will be proficient in an open-source operating system- they may perhaps, represent a tiny percentage of the population, similar to farmers today compared to a century ago. Even if kernel programmers today are less than 1% of computer developers, kernel developers may comprise an even smaller percentage of programmers a century from now, because basic operation of systems may be stable for most applications. That said, new kernels for specific applications will likely still be developed. It is likely AI may comprise a larger percentage of augmented operations in a vehicle or machinery.

It might be hard to picture a car as an advanced computer, but the trend going that way suggests computers will rely less on microcontrollers and conventional user-space kernels for infotainment/navigation, and more on neural processing units. Software will be defined by hardware, but hardware needn’t be defined by software. Reprogrammable architecture can run on platforms that are not constrained by single purpose applications.

Plato’s *Symposium and Phaedrus* was a dialogue that suggested writing was a technology that could displace oral teachings. While that did not occur, suppose one day someone suggested that software was obsolete- that all of the useful kernels could be loaded from a library and no one would ever need to write a new software. A century could pass and no new software is needed, because most ailments have been cured or treated, and emerging diseases become rare (or at least software isn’t invested in to try to cure it).

In a similar way, writing as a technology hasn’t changed its essential nature, despite the printing press and digital publishing. “Writing” still involves constructing words out of text, whether it is a stone tablet or a digital tablet. It has become more portable, and almost like a commodity. In this way, writing is a currency, but a technology nonetheless. Kernels for average users is less a currency and more of a commodity- it is traded as a technology but its internals, like a car, are not examined (except by appraisers and estimators).

The kernel as a car engine analogy is certainly not new. However, computing increasingly is finding its way into major appliances and vehicles in ways that are beyond real-time operating systems. Even as

linux begins integrating hard real-time operating systems, cars that use autonomous driving modes rely on open systems that are not Turing complete. In theory, a car that had a simple itinerary, such as a closed-course 0-60 test drive without external feedback, it could run on Prolog. But since cars need to respond to their environment real world tests, they need to be constantly improved upon when new data shows room for changes. It is possible that a century from now, cars will have supercomputers in them that are 100,000x -100 trillion times faster than they are now, but the cost to run those cars in electricity might allocate a larger percentage of the cars battery/energy storage system towards computing the safest path. A kernel for an automotive engine is likely to be a polycephalous neural net- an octopus with 8 brains that has redundant fail-safe mechanisms to avoid collisions. It might even be called *Octowheels*, or *Oct-on-wheels. High octane fun.* Software is technology. Cars are technology. Cars are not software. But one day that distinction may be much harder to notice.