The Stability After Confusion

Richard P. Gabriel

Software and software systems are built to exacting precision. We have designed our programming languages in support of this and we blanch at the idea that a program could work that was not 100% precise and formal. A house of cards falls because each piece needs to exhibit only a small range of perturbations to remain stable and robust. Our ideas for programming languages come from hard-edged bits, super-rational abstractions, and rigor of perfection over the poetry of imperfection.

But, biological systems are made of many simple but complexly interacting subsystems, and many of the mechanisms in them are for the purpose of maintaining the life and integrity of the system as a whole. Fluids carry concentrations, individual components—cells for example—die and are replaced (more or less), statistics are more important than individual signals, and so on.

Therefore, I believe that software of the (near, I hope) future will not be purely formal, not be entirely built on signals and bits, will use proximity to express intimacy of computation, will use evocation over prescription, and will layer detailed, desired, and precise functionality on a matrix of self-sufficiency and robustness, just as our thoughtful minds are hosted upon a meaty and persistently striving substrate.