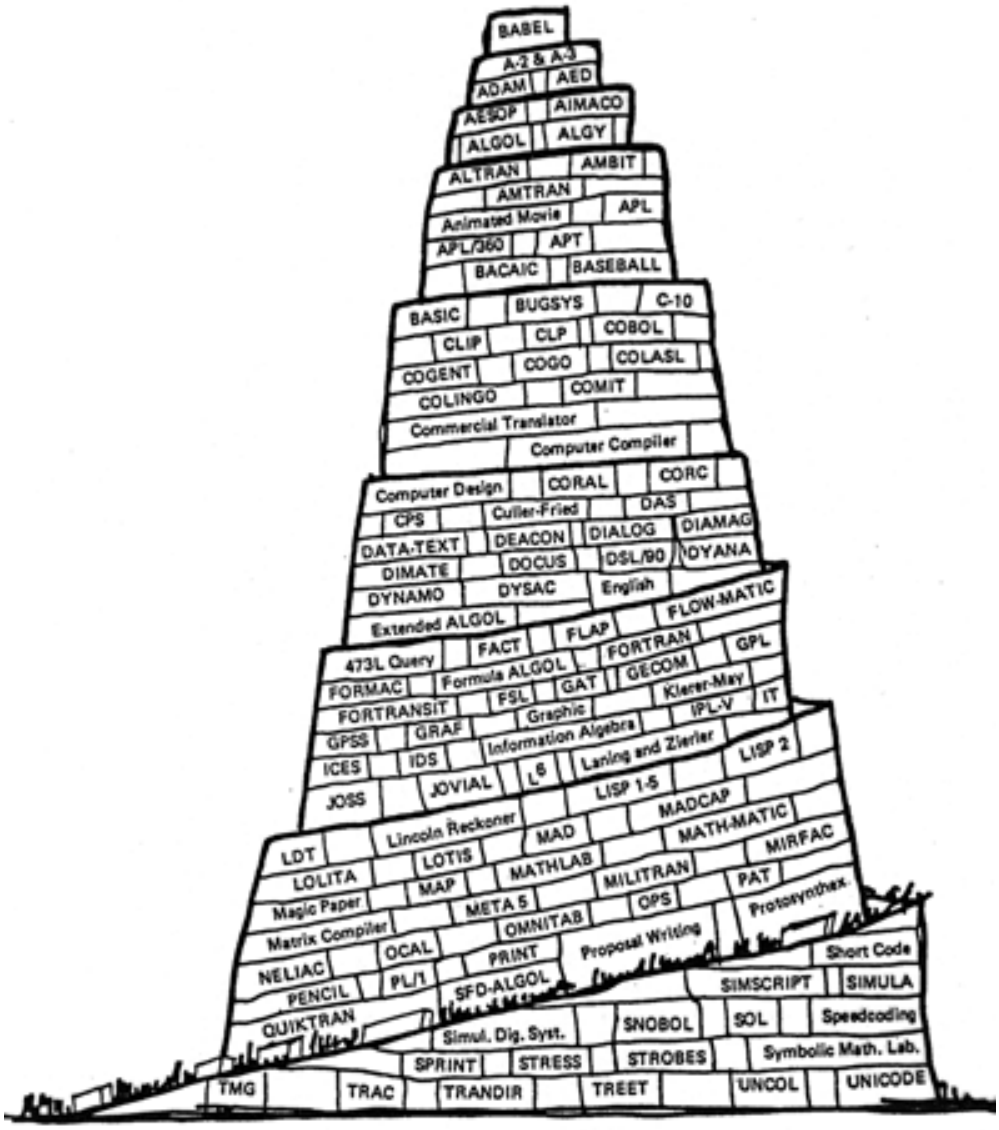
ENGINEERING COMMENTS ON SOFTWARE USING C-based languages AND OOP, Includes C, C++. C#, Java, Python, etc. (circa 1980 - 2015)



Software Quotes

"Continued and rapid growth in the power of hardware has not only enabled new applications and capabilities, but has permitted sloppy, unprofessional programming to become the virtual standard of business and industry. Hardware has allowed the software profession to avoid growing up, to remain in an irresponsible adolescence in which unstable products with hundreds of thousands of bugs are shipped and sold en masse." -- Larry Constantine [5]

In the first sentence of the preface of their book, **The C PROGRAMMING LANGUAGE,** [2], Kernighan and Ritchie state that "C is a general-purpose programming language which features economy of expression, ... C is not a 'very high level' language, nor a 'big' one, ..." In the second paragraph of CHAPTER 0: INTRODUCTION, it states that "C is a relatively 'low level' language."

C is quirky, flawed, and an enormous success.
- Dennis Ritchie, one of the original C authors - from van der Linden's Deep C Secrets, [3].

To quote a "quote of the father of C++ ", from Deep C Secrets, [3]:

C makes it easy to shoot yourself in the foot. C++ makes it harder, but when you do, it blows away your whole leg. - Bjarne Stroustrup

Apart from the OO concept, there are many reasons why C-based languages are difficult to understand.† We refer again to van der Linden's book**,** [3], on the use of C and C++ in a production environment at SUN, where he questions the placement of the burden of translating code. *Should it be on the programmer, or on the language translator?* On page 64, he states:

> "C's declaration syntax is trivial for a compiler (or a compiler-writer) to process, but hard for the average programmer. Language designers are only human, and mistakes will be made. For example, the Ada language reference manual gives an ambiguous grammar for Ada in an appendix at the back. Ambiguity is a very undesirable property of a programming language grammar, as it significantly complicates the job of a compiler writer. But the syntax of C declarations is a truly horrible mess that permeates the *use* of the entire language. It's no exaggeration to say that C is significantly and needlessly complicated because of the awkward manner of combining types."

The programming productivity problem was highlighted by Paul Strassmann, former Assistant Secretary of Defense for C3I, at a 1992 Ada Symposium at George Mason University, see [4]. There he described the necessary transition of the software industry in his speech "From a Craft to an Industry." He presented the results of a study on the resistance to change in the mode of production of software by what he termed the "loner programmers," the people that every computer installation has come to depend on. He said:

> "You can easily identify them. ... They are immersed in their craft, but find it difficult to explain or document it. They usually work late into the night, trying to fix a problem caused by low quality and frequently repaired incomprehensible software. ... They place little reliance on assistance from others and most likely disregard orderly documentation and business practices... The computer code they write is unique, elegant, and usually incomprehensible to others - which explains why they are highly valued as indispensable staff."

As software complexity has grown, current approaches to computer programming have taken a major toll on run-time speed. For over two decades Moore's Law doubled processor clock rates every 18 months, helping to conceal the underlying software problems that led to (Niklaus) Wirth's Law: "Software gets slower faster than hardware gets faster." Today we are faced with the realization that semi-conductor expert Jim Meindl's prophecy was accurate, [5]: the Moore's curve for speed has flattened. The doubling of CPU clock rates every 18 months is gone. To make up for Wirth's law, manufacturers are building multi-core processor chips, forcing the use of parallel processing.

---

† A common joke is that C++ is a *write-only* language. One programmer writes it and no one else can read it.

Separation of Skills - The Requirement of an Industrial Approach

We submit that separation of skills is the biggest differentiator between a craft and an industry. The industrial revolution not only automated many jobs, it took crafts and turned them into industries. This was most apparent in factories, where different job skills were clearly classified. One did not have to be a craftsman to participate in the production of goods. One could look to a career path that moved up the line as one increased architectural or management skills. But such an environment does not exist in software. And this is what is keeping software from moving from a craft to an industry.

There is a lack of production-oriented technology in software to support separation of the skills. In other words, an environment must exist that supports the separation of skill sets. With everyone using the same set of tools there are no differences. In every other engineering discipline, there is a clear separation of designer from technician, architect from builder, etc. So why not in software? Because there have been no tools to support this separation.

In a Software Special Report, [6], Business Week posed the question "Can the U.S. Stay Ahead in Software?" This article described the growing number of software engineers and programmers in other countries whose price per hour is less than 1/3 of their equivalent in this country. Thus, the cost of software development in foreign companies could be much less than their U.S. counterparts, even if they were not nearly as efficient at it.

In that same article, Lim Joo-Hong, deputy director of research at Singapore's NCB brought out one of the major factors - "Software only needs people. There is little need for a lot of other resources." In that same article, Edward Yourdon, publisher of the monthly newsletter *American Programmer* in New York warned that cheap labor abroad could begin to make low-level programming jobs in the U.S. obsolete. He was quoted as saying "The only thing that has prevented it from becoming a crisis so far is that the software industry is growing so fast that we haven't seen many jobs taken away." The article further warned "Without such entry-level jobs, the U.S. won't be able to employ large numbers of computer science graduates, further discouraging careers in the field."

But look again. While hardware engineers produce great feats, tearing down barrier after barrier, looming problems in software are hiding behind them. Quoting Marcus Ranum, [7],

> "...I see that Microsoft, Intel, and AMD have jointly announced a new partnership to help prevent buffer overflows using hardware controls. In other words, the software quality problem has gotten so bad that the hardware guys are trying to solve it, too. Never mind that lots of processor and memory-management units are capable of marking pages as nonexecutable; it just seems backward to me that we're trying to solve what is fundamentally a software problem using hardware. It's not even a generic software problem; it's a runtime environment issue that's specific to a particular programming language."

But Ranum's article is just one example. There is a large group of people experienced in both sides of the computer field - hardware and software - saying the same thing. More importantly, the statistics on productivity show that the software industry has been going downhill every year. But that is not the worst of it. To take advantage of a large number of parallel processors requires a new approach to operating systems. And here, it seems that we can't even get it right for a single processor.

Articles about Microsoft's problems with its new Vista operating system bear this out. In the Sept. 2005 Wall Street Journal, Guth, [8], describes Microsoft's delays in its effort to come out with Vista, a new version of the Windows operating system. It quotes Microsoft executives saying that there was no architecture!

This article was followed up by Cusumano in the ACM, [9], where he talks about the gridlock occurring on the Vista project, stating:

> "We now know that the chaotic 'spaghetti' architecture of Windows ... was one of the major reasons for this gridlock. Making even small changes in one part of the product led to unpredictable and destabilizing consequences in other parts since most of the components were tied together in complex and unpredictable ways."

It now appears that we are already into an era where the computer field is constrained by software problems that present barriers to using new hardware technology. To counter this problem, we must do a reversal on our approach to developing and supporting software. And this approach must be based upon a new paradigm that takes full advantage of all that hardware.

As described in *Microcosm* by George Gilder, [10], human inertia is the major deterrent to innovation. A university teacher of programming characterized the new technology presented in VisiSoft as "unprofessional." When asked why he considered it unprofessional he replied "*Anyone can use it!"*

REFERENCES

1. Constantine, Larry, "Back to the Future," Communications of the ACM, Vol 44, No. 3, March, 2001.
2. Kernighan, B.W., and D.M. Ritchie, *The C PROGRAMMING LANGUAGE,* Prentice Hall, Englewood Cliffs, NJ, 1973.
3. van der Linden, P, *Expert C Programming - Deep C Secrets,* SunSoft Press - Prentice Hall, Englewood Cliffs, NJ, 1994.
4. Strassmann, Paul, "From a Craft to an Industry," Ada Symposium, George Mason University, 1992.
5. Meindl, James D., Keynote Address, 1997 Hot Chips IX Symposium, Stanford Un., Palo Alto, CA.
6. "Can the U.S. Stay Ahead in Software?", Special Software Report, Business Week, McGraw-Hill, March 11, 1991.
7. Ranum, Marcus J., *SECURITY - The root of the problem*, ACM QUEUE, June 2004.
8. Guth, Robert A., "Battling Google, Microsoft Changes How It Builds Software," The Wall Street Journal, Sept 23, 2005, page 1, column 5.
9. Cusumano, Michael A., "What Road Ahead for Microsoft and Windows?," Communications of the ACM, July 2006, pg 21-23.
10. Gilder, George, *Microcosm,* Simon and Shuster, New York, NY, 1989