# *Is Silicon Valley Headed Toward Detroit?*

***Or Ready to Benefit from Software CAD Technology***

**William Cave† - March 12, 2015**

## The Need For Speed

Starting with the atomic bomb, computer technology has been driven by the need for speed. An all-electronic version of mechanical calculators - the ENIAC - solved part of this problem. But programmers still had to wire boards to create instructions to add, multiply, and manipulate the data, and had to account for circuit delays and race conditions.

The complexity of the hydrogen bomb led to the first stored program computer - the MANIAC, where instructions were stored in memory along with data. Since instructions were accessed *sequentially*, they were *independent*. Programmers no longer worried about timing or synchronization, which was shifted to the logic designers of the machine. Computer programming became simple. But the key to speed was memory which has increased from a few Kilobytes on the first machines to about 10 Terabytes on today's PCs.

From 1982 to 2006, software application speeds doubled every 18 months - aided simply by increases in computer clock speeds. Then chip power dissipation densities forced clock rates to level off. To meet the need for speed, hardware manufacturers have put multiple processors (cores) on a chip, forcing programmers to decompose a task into parallel parts. But when software must be split across multiple processors running concurrently, the great simplification of Von Neumann's instruction set architecture - the independence of sequential instructions - no longer applies. Various people have invented new architectures, but every time someone tries to change this paradigm, programmers get hit with the synchronization problem.

## Changes in Software Productivity

Prior to the 1980s, software productivity increased rapidly. Since the late 1980s, thanks to the move to C-based languages and OOP, software productivity has gone down - *faster than any other industry in the U.S.* (see charts below). Whereas computer chips had the highest productivity growth of all industries in the 5 years prior to 1995 (chart 1), software productivity growth was negative (red bar). Recent studies make it clear that this situation hasn't changed. In the year ending 2004 (chart 2), computer chips still had the highest productivity growth while software still had the most negative growth (red bars). *This is before parallel processors changed the market.* The high cost of building and maintaining software, and large number of project failures has put many projects on hold. Large companies are now outsourcing software projects overseas to India, China, and similar countries.

The most recent exposure of this problem is described in the February 21, 2015 issue of the Economist magazine. The underlying cause is the same job protection mentality that occurred on shop floors in the automobile industry in the 1960s, recently putting Detroit into bankruptcy. As discussed below, this is putting Silicon Valley on the same course as Detroit.

† The author is with Visual Software International - www.VisiSoft.com

CHART 1. Data From Business Week - January 1995.

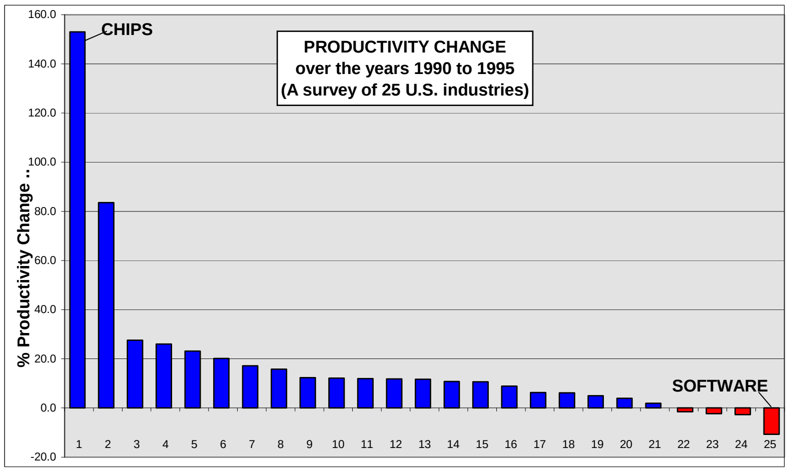
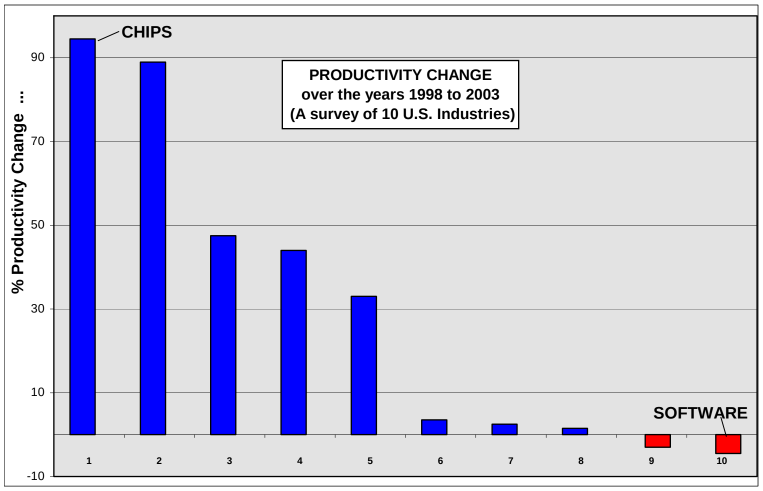


CHART 2. Data From Groth, IEEE Software, November/December 2004.



## Hardware Manufacturers Caught In The Middle

As indicated above, increases in software application speeds were due to computer clock speeds doubling every 18 months. This concealed problems that led to (Niklaus) Wirth's Law: "Software gets slower faster than hardware gets faster." But by 2006, computer clock rates had leveled off. To keep the speed curve headed up, hardware manufacturers started putting multiple processors (cores) on a chip, leading one to believe that having 8 processors on a chip would speed up applications by a factor close to 8. Faced with the problem of decomposing a task into separate parallel parts, the benefits of sequential instruction independence are gone, and the problem of timing and synchronization is back. Programmers must now split software across multiple processors running concurrently, a task not supported by popular software languages.

This new level of complexity has programmers looking back to the chip designers to solve their problems by moving special software functions into the hardware. The chip space wasted trying to accommodate these false needs is huge. *This is going in the wrong direction!*

## The Right Direction For Change

Enter the age of Computer-Aided Design (CAD). VisiSoft - a CAD system for software - has been providing huge simplifications for building complex systems for many years. *Understandability* is a critical aspect of software productivity. Linearization of complexity requires that the system be decomposed into modules that are maximally *independent*.

Implementing understandability and independence depends directly on the language used to produce the code. It must be highly readable by subject area experts and easily support substantial layers of hierarchical structures of both data and instructions. Current languages have none of these properties.

## Resistance To A Disruptive Technology *(The Real Software Problem)*

VisiSoft is a disruptive technology that will change the underpinnings of the software world for decades to come: (1) using this CAD system, subject area experts can build software directly without programmers; (2) it provides huge economic benefits to both developers and end users. It takes less time to build software for a parallel processor than current approaches on a single processor, being cut by factors of 2 to 5. Books by Clayton Christensen and Thomas Kuhn describe the resistance that must be overcome when introducing disruptive technologies:

- Job Security - People with years of experience in the current technology see the loss of value of their expertise (*young people love VisiSoft; they learn it fast*).
- Financial Competition - Huge investments in current technology are at risk of being wiped out (*there are new ones to be made using VisiSoft*).
- Not Invented Here (NIH) - People in government/academic research must justify large salaries they receive to develop such technologies (*they can be the early evaluators*).

Having made investments in becoming proficient in a subject, one does not want to think of that time as wasted. A university teacher of programming characterized VisiSoft technology as “unprofessional.” When asked why he replied “*Anyone can use it!* ” Proficient C-based language programmers will not use a disruptive technology. They are a significant barrier to change. As described by George Gilder, ***human inertia is the major deterrent to innovation***.

## Comparing Software To The Automobile Industry

On a Friday afternoon, two young computer engineers in an Austin Healey were headed from Ann Arbor, MI to Detroit. On the way they had a business meeting with a computer manufacturer. Before the meeting, someone said that company policy did not allow foreign cars on the lot - the car had to be moved. After the meeting, the engineers asked about places to go in Detroit. They were told to rethink driving in the city with the Healey - it was definitely dangerous. They could have their tires slashed and the leather top cut off - while sitting in the car. Someone provided directions to a parking lot in the city that hid foreign cars, indicating that, if they could get there, it would be safe.

At that time, new technology companies were building robots for factory automation. But these were banned from the shop floors of U.S. auto manufacturers. Even quality control experts - Joseph Juran and J. Edwards Deming - were banned from taking measurements to improve quality and lower costs on the assembly lines, so they went to Japan. All of this was justified based upon *job security*. To get elected, politicians stood firmly behind this policy.

But when buyers who control their economic choices are driven by their own hard-earned money, the ball game becomes fair. Those who deliver the best results are the winners. Fifty years later Detroit is in bankruptcy, devastated by foreign car manufacturers.

One would believe this kind of thinking to be unacceptable in a technology field like computers. Yet thanks to the integrated circuit chip, the computer field was split into hardware and software. Hardware engineers use CAD tools to design chips, and take measurements to compare designs. Software is more of an art form, particularly from a user standpoint. Who are the major buyers? They are probably part of the fast growing social networking market - texting to their friends, kids, and grandkids.

In nations with low labor rates, products produced by low skilled jobs will be very competitive. Where labor rates are high, low skilled jobs are hard to support and one must look to sources of revenue requiring advanced capabilities. Since the U.S. has been a major developer of high technology, it should be striving to dominate markets that demand a high technology edge, one that can be maintained for the long haul. The auto industry is one of those markets. As Japan's labor rates exceeded those of the U.S., they used shop automation - everywhere!

## Achieving Real Job Security - By Seeking the Truth

The markets depending upon computers and automation are growing rapidly, a trend that should continue for decades, and an area in which the U.S. excelled for many years. However, *since the late 1980s, productivity in the software field has dropped - faster than any other industry*. The high cost of building and maintaining software, and the large number of project failures has put many projects on hold. Companies are outsourcing their software overseas.

The underlying cause of this problem is hidden from public knowledge. It is the result of the same job protection mechanisms that dominated the U.S. automobile industry in the 1960s. This time it is the supposed "high-tech" people (the programmers) who falsely believe they are protecting their jobs from lower skilled people, when they could dramatically improve their own productivity using CAD systems. But, *they refuse to take measurements and make comparisons*.

This problem occurred in the U.S. software field in the late 1960s when programmers insisted that business software had to be written in assembly language. But at that time, software managers came up through the ranks and understood what was going on (job protection). These managers made the decision to switch to a new programming language (COBOL) where high school graduates could produce software faster than PhDs in mathematics using assembly language. Using COBOL, the U.S. software field expanded dramatically.

Today's programmers do not aspire to management, so management does not understand what programmers do. The productivity of tools used to build software has regressed dramatically since the 1980s. Programmers hide what they build behind difficult to understand languages, protecting their jobs by ensuring that only they know what's in the code. Now Silicon Valley is following the same blind path as the programmers, trying to put into hardware knowledge that only exists inside a specific application. Lack of knowledge of the overall problem is causing hardware designers to head in the wrong direction - *to preserve the current approach to building software* - the root cause of the problem. Watching this unfold is hard to believe. Led by programmers, is Silicon Valley headed down the same path as Detroit?

The software field is a clear case where new technology can be used to dramatically improve productivity in a very desirable job market. *It's time to seek the truth and take the data!*