Why Hasn't Software Productivity Improved?

Don Anselmo - Presented at the* Software Summit, *Washington, D.C., May, 4, 2004

In a September 1991 Business week article titled "Software Made Simple," [1], industry experts offered great hope for productivity gains in the future with Object Oriented Programming (OOP) technology. The article admitted that there were naysayers who compared OOP to the promises of Artificial Intelligence (AI). But it was stated that, unlike AI, object technology would have an immediate and practical payoff.

In the July 1996 issue of Software Developer & Publisher magazine, an article titled "Software Survival," [2], analyzed the reasons for declining software productivity. It quoted two other articles, [3] and [4], that showed, while productivity in the computer hardware industry was *increasing* faster than in any other industry, software industry productivity was *declining* faster than all others for the period 1990 - 1995. More recently, Business Week, [5], tabulated their findings in industry productivity and showed that software productivity finished dead last, declining from 1998 to 2003.

As we approach what *appear* to be physical limits in the semiconductor industry, engineers have continually removed the barriers. Through the use of Computer-Aided Design (CAD) tools with graphical visualizations, the promise of Moore's law has continued to be fulfilled. Microprocessor speed, memory density, and affordability have skyrocketed. As stated by Larry Constantine, software engineering has capitalized on these advances to offset its own poor productivity. But unlike hardware, software has never capitalized on the underlying engineering concepts to improve productivity.

The tools of software engineering are at the level they were at in the 1960s when a one megahertz clock and 64 kilobytes of memory were the norm. With many gigabytes of memory and many gigahertz in processor speeds available to a programmer, it is not surprising that the old tools and methods still suffice. But imagine trying to build today's computers with the manual layout tools available in the 1960s.

In a recent talk at the Software Industry Workshop in Washington D.C., Dr. A. La Salle, [6], posed the questions: " Is the production of software in the USA somehow defective? If it is defective, how did it get that way? Can we do anything about it?" I believe the answers to the first and third questions are Yes and Yes. As for the middle question, many people argue that C, C++, Java, and OOP are the root of the problem. [7], [8], [9].

In the feature article of a 1994 issue of Upside, [9], five leading technologists were interviewed to get their view on the world of technology in the new millennium. The questions covered broad areas of communications and automation. One of the questions was "What advancement will be the biggest disappointment?" Surprisingly, Gordon Bell, architect of DEC's VAX family, and John Warnock, CEO of Adobe Systems had the same answer - Object-Oriented Programming! Warnock said, "I think the whole object thing is a red herring."

Maybe the origins of the C language can shed light on the issue, see references [10] through [15]. In 1969, the MULTICS project (a collaborative effort between Bell Laboratories, GE and Project Mac at MIT) was cancelled leaving members of the Bell Labs staff without a project. The purpose of MULTICS was to produce a timesharing environment to replace the cumbersome batch environments in use at that time. Without such a facility, the Bell researchers were deprived of a convenient environment in which to share their work.

In his spare time, MULTICS team member Ken Thompson had developed a computer game called "Space Travel" that simulated the motion of planets in the solar system. A player could cruise between planets, attempting to land the ship. First written on Multics and then rewritten in FORTRAN for the GECOS operating system on a GE 635 computer, the game's display was jerky and hard to control because the player had to type commands to control the ship. Also, it cost about $75 in CPU time on the GE 635. Thompson soon found a little-used PDP-7 computer with an excellent display terminal, and with help from Dennis Ritchie, Space Travel was ported to this machine. This was written in assembly language for a cross-assembler that ran under GECOS and produced paper tapes to be carried to the PDP-7. Though it made a very attractive game, it was difficult to prepare programs for the PDP-7.

Thompson then set about to write a FORTRAN compiler for the machine but quickly abandoned that effort as too ambitious. With his initial motivation being the port of Space Travel, he started writing a translator for a language he called B. B was based upon the word-oriented Basic Combined Programming Language (BCPL) developed in the UK. His principal concerns were: (1) to use a Spartan syntax to keep the compiler simple to write; and (2) to keep the compiler small to fit into the PDP-7's tiny memory. The compiler served its purpose but produced interpreted code that was inherently slow.

The PDP-7 became a natural candidate to develop what was to become the UNIX operating system, and development proceeded in the PDP-7's bare assembler - with no loader or link-editor. This led to an effort by Thompson, Dennis Ritchie, and others to design a file system as the basis for a new operating system. At this point there was no corporate objective or plan to develop a product. This was a project intended to create a "friendly" environment in which a small group of researchers could share their work.

In 1971, UNIX was moved to a PDP-11, and although the first version of UNIX was written in assembler, Thompson's intention was that it be written in his new language. Ritchie added new types to B, which for a while was called NB for "New B."

After a mandate from the patent department to create a "file system" for handling patents, Ritchie decided to modify B and produce the C language, [10]. To make it fast enough, C would not be interpreted. Also, since the word-orientation of B did not match the new byte-oriented machines, e.g., the PDP-11, new data types were added that were byte oriented. The desired file system was then written in C. With the provision of file manipulation features and a shell, it became the first version of UNIX.

Victor Vyssotsky, who had been the head of the MULTICS project at Bell Labs (later Executive Director of Research in the Information Systems Division of AT&T Bell Labs) stated, “When UNIX and C evolved within Bell Labs it was not the result of some deliberate Management initiative”, [11]. At that time programmer and software productivity were guided by the hierarchy of Labs managers rather than by a serious effort to provide high productivity tools to improve the speed and output of these projects.

The Laboratories later recognized the need for tools to produce large software systems and developed PWB/UNIX, which was used in #5 ESS (a major network switch) for development of the Operations Service System (OSS) software. OSS was used by the baby bells and other clients buying switches. After the breakup, it surfaced that this software was unreliable and difficult to maintain. This resulted in a growing loss of switch sales to other switch manufacturers e.g., Northern Telecom.

Probably the most important force behind this revolution was the desire on the part of Bob Allen, Chairman of AT&T, to take on IBM in the computer business. The fact that Bell Labs had already built its own computers, and was now embarking on the new UNIX operating system provided the impetus to proceed with a big push. Over the next few years, AT&T pumped huge investments into UNIX (some say Billions), and C went along for the ride. Part of this ride was the invention of Object Oriented Programming, and the lead of Bjarne Stroustrup, and many others from Bell Labs, in publishing papers around the world. The rest is history. It has replaced the prior history of software, which is now forgotten.

It is time to take a hard look at the tools and methodologies in current use. Many people joke about the inertia in the software field. But, as programming jobs move overseas, we are now being forced to take a fresh look at new and existing approaches (some have been rejected or discouraged for “political” reasons). This would be a pertinent topic for discussion at the Center For National Software Studies next meeting May 10-12, 2004 in Washington D.C.

References

1. *Software Made Simple*, Business Week Cover Story, McGraw-Hill, September 30, 1991, pp 92-100.
2. *Software Survival*, Software Developer & Publisher, W. Cave, July/Aug1996.
3. *PROGNOSIS '95*, Business Week, McGraw-Hill, January 9, 1995, pp 72-80.
4. *Chaos, Charting the Seas of Information Technology*, The Standish Group International Inc. Report, Dennis, MA, 1995.
5. *Industry Outlook*, Business Week, McGraw-Hill, January 12, 2004, pp 92-100.
6. *Software Industry and Economic Security*, Dr. Anita J. La Salle, Software Industry Workshop, April 27, 2000.
7. *The History of Computer Programming Languages*, Steve Ferguson http://www.princeton.edu/~ferguson/adw/programming_languages.shtml
8. *The Emperor with No Clothes*, Henry F. Ledgard, Communications of the ACM, Oct 2001.
9. *Musings on the Millennium*, Feature Editorial, Upside Magazine, October 1999.
10. *Development of the C Language*, Dennis M. Ritchie, Second History of Programming Conf., 1993, http://cm.bell-labs.com/cm/cs/who/dmr/chist.html
11. *On the Early History and Impact of Unix- Tools to Build the Tools for a New Millennium*, Netizens: An Anthology, Chap 9, http://www.columbia.edu/~rh120/ch106.x09
12. *The Unix Oral History Project*, Michael S. Mahoney, http://www.princeton.edu/~mike/expotape.htm
13. **The C Programming Language**, Kernighan and Ritchie, Prentice Hall, 1978.
14. *The Unix Timesharing System*, BSTJ, Vol. 57 No.6, July- August 1978.
15. *The Unix System*, ATT/BTL Technical Journal, Vol. 63 No.8, Oct. 1984.
16. *A Tale of Three Disciplines and a Revolution*, Jesse H. Poore, IEEE Computer Society, Jan 2004.
17. *Measuring Productivity in The Software Industry*, Donald Anselmo and Henry Ledgard, Communications of the ACM, Vol. 46. No.11, Nov 2003.

BRIEF RESUME OF DON ANSELMO (Dec 1935 - Nov 2004)

EDUCATION

BS, Electrical Engineering - VPI
MS, Electrical Engineering - NYU
CDT Program, Bell Telephone Laboratories
Post Grad, Mathematics - Stevens Institute

HONORS

ETA KAPPA NU
PHI KAPPA PHI
TAU BETA PI

GENERAL BACKGROUND

Director Interconnection Technology, ATT Bell Labs
Director of Computer Development, ATT Bell Labs †
Vice President & Director of Product Management, ATT Technologies ††
President and COO, Open Connect Systems Corp.
Vice President, Technical Systems Division - Motorola Computer Group
Director, Telephony Products Office - Satellite Communications, Motorola
Director, Strategic Program Development, Network Solutions Sector, Motorola

† As Director of Computer Development, Bell Labs: Responsible for the development of 3B computer line and the UNIX operating system, contributing to the commercialization of UNIX as an ATT product.

†† As Director of Product Management, ATT Technologies: Oversaw product management for the 3B computer line, ATT PC, and Olivetti relationship for 3B distribution in Europe. Controlled management of corporate investments by ATT in several third party strategic relationships. Part of the board membership representing ATT in strategic investments.