Software Industry vs. Software Society: Who Wins in 2020?

An STS Forum Position Paper by Michael Tiemann
President Open Source Initiative
Vice President Open Source Affairs, Red Hat

Commercial software is not yet 60 years old, and in that time it has created a new economic sector and created some of the wealthiest people on earth. ICT is now ubiquitous in the developed world, and spending on combined hardware, software, and ICT-related services accounted for $1T USD in 2004 [1]. But not all of the IT industry's indicators are healthy: the largest companies are struggling to grow and **quality problems account for at least $180B USD/year in ICT write-offs** [1] (more likely at least $286B USD when "challenged" projects are included). Compared to $610B USD profit earned by the Fortune 500 on $9.1T USD revenues in 2005 [2], the dead loss from IT write-offs is astonishing, and could explain why **more than 90% of the leading IT vendors fail to achieve either a "good" or "excellent" rating for value from at least 80% of their top customers** [3]. Looking to the future, can an industry sustain quality-driven losses of 18%-29% of total revenues?

The documented failure of industry as a model for software is almost as old as the software industry itself. IBM's OS 360 project, begun in 1961, was perhaps the most ambitious attempt to industrialize software. Fred Brooks, the IBM executive responsible for the project [4], hired the best coders, managers, analysts—whatever it took. The product shipped—barely—in 1965, but Brooks wasn't finished until 1975 when he wrote the book *The Mythical Man Month*. The title comes from his insight that "it takes a woman nine months to have a child, no matter how many women are assigned the task." This was the first clue that the process of industrialization—breaking down tasks into simpler steps, optimizing the steps, and then managing the steps through production—had very significant limits when applied to software. Today, this book is often the first reference given to explain why a multi-billion dollar software project will slip yet again, run over budget again, and why it will ultimately ship without the very features that justified the project in the first place.

While IBM and others struggled to industrialize software, the mainstream industrial model itself was beginning to show weakness. In the 1940s, Dr. W. Edward Deming properly identified serious problems with the model that focused on optimization in terms of measurable (but arbitrary) inputs and outputs as related to costs and production. Deming had no luck convincing American companies to focus on quality as defined by the customer, nor to recognize the value of people working in teams on a larger and transforming goal, nor to consider whether work gave people joy. Fortunately Deming found a receptive audience in Japan, and the transformation that resulted has set the standard for modern industry, lowering costs, accelerating innovation, reducing environmental impact, improving safety, and inspiring pride in workmanship [5].

By the 1980s, Deming's work had reached even those still trying to industrialize software, in the context of the Software Engineering Institute (SEI) and their Capability Maturity Model (CMM) in 1987 [6]. First adopted not in the US, but offshore, in Chennai, India [7], it was credited with dropping software defect density 98% while doubling profits [7]. Though these first attempts have now been retired [8], the urgent question remains: why is software quality so low [9]? Why is it then that companies like Microsoft spend fully a third of their R&D budget correcting remedial security problems that should never have existed in the first place [10]? As President of the Open Source Initiative and Vice President of Open Source Affairs for Red Hat, I believe software quality is low because companies focus more on control than on innovation or quality.

The most significant transformation in ICT is the emergence of Free and Open Source Software (FOSS). Consider the popular internet as embodied by the World Wide Web. This vision was first published by Vannevar Bush in 1945 [11], and prior to 1990 there had been dozens of attempts to create such systems, the most popular being the French MiniTel system [12]. But none of these systems became truly ubiquitous until the underlying software was free as in freedom. The freedom to read, modify, and share web server and client code led to an explosion of innovation, and to date the most popular product of that explosion, the Apache web server, remains dominant in spite of considerable efforts by some powerful companies to take over that space [13].

This came as no surprise to Tim Berners-Lee, creator of the World Wide Web, who explained his decision to make the original web software free and open:

> [H]ad the technology been proprietary, and in my total control, it would probably not have taken off. **The decision to make the Web an open system was necessary for it to be universal.** You can't propose that something be a universal space and at the same time keep control of it. [14] – Tim Berners-Lee

The popular internet has created a dramatic new factor in the the software industry equation, which is that **developers and users represent a creative continuum**, not distinct castes. Developers can be any people interested in a problem, not merely people employed to work on a specific problem. Free and Open Source software turns on its head the practice of optimizing software production by limiting the number of people working on a problem. Instead, those employed (producers), those annoyed (customers) and those who merely enjoy working on a problem can work together in a problem-specific, rather than organization-specific or property-specific way. In his book *Democratizing Innovation*, Eric Von Hippel presents several studies which, together, show that 85% of all quantum innovation (as opposed to incremental innovation) is user-driven [15]. **The proprietary software model therefore limits innovation to 1/6th of its theoretical potential.**

The democratization of innovation has also demonstrated a remarkable solution to the problem of *The Mythical Man Month*, thereby transcending the limits of conventional industrialization. For example, *sourceforge.net* is an open source development resource that hosts over 100,000 projects and has more than 1.2M registered users [16]. Extrapolating from the extensive FLOSS survey of 2002 [17] (and updated in 2005 [18]) there are over 490,000 *sourceforge.net* developers who spend more than 10 hours a week or more tending their open source projects [19]—an aggregate effort of some 5 million person-hours per week. The three top reasons they list for their involvement is:

1. Because it's fun
2. Because it improves their skills
3. Because it is good for society

Note that this does not include Linux developers (who use *kernel.org*, not *sourceforge.net*), nor Apache, nor the GNU project, nor many of the other larger and more heavily commercialized open source projects. To put these 5 million joy-filled person-hours per week into perspective (again, this does not include Linux, Apache, GNU, or many of the other "large" projects), let's look at the productivity potential of the most successful proprietary software company, Microsoft, in two ways:

1. If all 61,000 employees [20] wrote code, they would have to work over 80 hours/week
2. If Microsoft's $6.6B/year R&D budget [21] were spent on programmers averaging just $25/hour, they could pay for about 5 million person-hours of work per week

Thus, the *sourceforge.net* website has equaled or exceeded Microsoft's productive potential using a social, not an industrial model. When we consider all the open source developers not included in the *sourceforge.net* numbers (numbers that are increasing exponentially), we see the clear emergence of a new software production capacity entirely outside the conventional limits of the industrial model. Now, as we look to the year 2020, is this new capacity a threat or an opportunity?

The emergence of this new production capacity has triggered the same kind of reaction from the entrenched status quo as did the appearance of high-quality products coming from Japan after the Second World War: fear, uncertainty, and doubt. However, those who have attempted to measure the results, rather than simply denying or disparaging them, have found precisely the kind of improvements that Deming would have predicted by taking a transformative approach.

According to findings published by Coverity [22,23,24], typical proprietary software has a defect density of 20-30 defects per 1,000 lines of code (KLOC), a number relatively unchanged since the 1960s. When they measured the quality of the Linux kernel (and later, other open source software) they found the following results:

2004: 985 defects in 5.7 MLOC of Linux kernel source code, or **99.3% lower defect density** than average (compare to 114,000 to 171,000 defects in same amount of code) [22]

2005: While the Linux kernel grew 4.7% in overall code size, defect density decreased by 2.2%. Moreover, **100% of all "serious" defects identified were fixed within 6 months** [23].

2006: The survey was expanded to entire LAMP stack and an additional 32 OSS programs. No correlation found between size and defect density, implying **OSS development methodology is not limited by scale**. [24]

What the top industrialists could not achieve with proprietary software and financial capital, free software has demonstrated with community development and intellectual capital.

But free and open source has done more than revolutionize the production or the quality of software. The open source model has opened innovation. Open source helped crack grand challenge science problems such as sequencing the human genome [25]. The world's largest public search engine, google.com, is built on open source [26,27,28]. Open source has become a fixture in tech-heavy disciplines such as financial services [29], military intelligence [30], online retailing [31], and next-generation cellular telephones and base stations [32]. And open source is helping to bridge the digital divide for millions of children who are literally off the grid [33]. The model has even informed the strategy of the Bill & Melinda Gates Foundation to better combat infectious diseases:

> [On July 20, 2006] the Bill & Melinda Gates Foundation announced that it would require that any researcher who accepts its grant monies for HIV/AIDS research will have to agree to share their scientific findings. The Gates Foundation was apparently frustrated that two decades of secrecy and competition among AIDS researchers have impeded efforts to come up with an AIDS vaccine. [34]

The battle of the next 10-15 years will be about who gets to control the ways in which software can be developed, sold, and used [35]. On the one side are those who argue that their profits are the only way to measure or deliver true social benefit. On the other side are those who strive for continuous improvement, forever and always, who believe that transformation is everybody's job— literally everybody. The software industrialists have massive financial capital and a 30 year head start on legislating proprietary interests over social interests. Those in the free and open source movements have the benefit of new intellectual capital born every day and a firm grasp of how increased, rather than decreased freedom improves innovation, quality, competition, and choice. Open-minded governments and the competitive private sector are taking note of the superior cost, quality, and performance demonstrated by open source. There will be no one winner or one loser, unless there is only one monopoly who ultimately dictates global IP policy through a single instrument and legal interpretation. Instead, the winners will be those who make the right choices— who look beyond an arbitrary model or a conventional understanding and find the truth.

[1] The Standish Group http://tinyurl.com/zbqdn
[2] Fortune 31/03/06: http://tinyurl.com/z4gmv
[3] http://www.cioinsight.com/article2/0,1397,1902404,00.asp
[4] http://www.cs.unc.edu/~brooks/FPB_BIO.CV.06.2006.pdf
[5] LA Times http://tinyurl.com/zs7y5
[6] SEI Technical Report http://tinyurl.com/gw6u7
[7] Business Week http://tinyurl.com/fe4su
[8] http://www.sei.cmu.edu/cmmi/adoption/migration.html
[9] http://www.cio.com/archive/101501/wasting.html
[10] Federal Computer Week http://tinyurl.com/juphw
[11] http://www.theatlantic.com/doc/194507/bush
[12] http://en.wikipedia.org/wiki/Minitel
[13] Netcraft Survey http://tinyurl.com/kq9l
[14] Tim Berners-Lee W3C http://tinyurl.com/eobkm
[15] http://web.mit.edu/evhippel/www/democ1.htm
[16] http://sourceforge.net/
[17] http://flossproject.org/floss1/stats_5.html

[18] OSCON 2005 http://www.flosspols.org/research.php
[19] http://flossproject.org/floss1/stats_7_2.html
[20] http://finance.yahoo.com/q/pr?s=MSFT
[21] http://finance.yahoo.com/q/is?s=MSFT&annual
[22] http://www.eweek.com/article2/0,1759,1741077,00.asp
[23] Internet News http://tinyurl.com/zuva2
[24] http://www.internetnews.com/stats/article.php/3589361
[25] http://www.ddj.com/184410424
[26] http://www.eweek.com/article2/0,1895,1877924,00.asp
[27] http://internetweek.cmp.com/lead/lead060100.htm
[28] Information Week 28/08/06 http://tinyurl.com/gdh2b
[29] eWeek 24/04/06 http://tinyurl.com/hou2z
[30] NSA http://www.nsa.gov/selinux/
[31] O'Reilly Feb 2004 http://tinyurl.com/z932r
[32] Linux Devices http://tinyurl.com/z4f7n
[33] http://laptop.org/
[34] http://onthecommons.org/node/941
[35] http://www.gnu.org/philosophy/lessig-fsfs-intro.html