How Open Source Software Can Save the ICT Industry One Trillion Dollars per Year

by Michael Tiemann
President Open Source Initiative
Vice President Open Source Affairs, Red Hat
November 1, 2009

Open Source software is an important[1,2] and growing[3] class of software. Open Source software is distinguished not by programming language, operating environment, nor application domain, but rather by the license(s) that governs the use, distribution, and, most importantly, the rights to access and modify the software's source code. Together, software source code, licensing, and community have dramatically changed many conventional assumptions about software and the software industry itself. And change is needed, because IT project failures are responsible for an estimated $1 trillion of dollars of outright waste and multiple trillions of indirect costs on top of that, which is unacceptable.

The $1T USD per year figure is extrapolated from the statistics that world-wide ICT is expected to top $3.4T USD in 2008[4] and because 18%-50% of all ICT investments, predominantly based on proprietary software because that is the dominant model of the industry today, are dead-loss write-offs.[5,6] An analysis also using the Standish data[7] argues that when indirect failures are added to write-offs, the world's GDP suffered ***a $6T USD loss due to IT failures in 2008,***[8] and uses the following example to support its case:

> Between 1994 and 2005, the Internal Revenue Service spent $185 million on a new electronic fraud detection system. The project was abandoned in 2006.
>
> According to a report in 2008 by the Treasury Inspector General for Tax Administration, the Federal Government lost approximately $894 million in fraudulent refunds during 2006 because the system was not operational.
>
> So in this case, the direct cost of the failure was $185 million and the indirect cost (the lost opportunity costs) was $894 million. Of course, that $894 million was lost just in 2006. Presumably the same amount was lost in 2007 and subsequent years.[9]

I have met and/or corresponded with perhaps a thousand CIOs, CTOs, and other executive-level IT managers around the world in the past five years, and while most of them do not have concrete data to back up the indirect failure measurement (due to lack of visibility to the numbers beyond IT), all agree that they themselves have observed *at least* the level and cost of failures that underlie the $386B USD dead-loss estimate that I first derived in 2006, and that now tops $1T USD per year in 2009. And this is not limited to US-only analysis: an article from India's Business Standard conservatively[10] estimates that India alone wastes $2B per year[11] on proprietary desktop operating systems and anti-virus software that are nevertheless overwhelmed by computer viruses and malware. Open source software has the potential to remedy this in two ways: by dramatically reducing the cost of software production, and simultaneously reducing the costs and risks of deployment and ongoing management over the entire IT lifecycle.

In 2004 it was estimated that the Linux kernel, if developed using proprietary methods and private investment, would have cost more than $600M to create, and that the roughly 1,000 packages typical in Linux distributions of the time would have cost more than $1B[12]. In 2005 an analysis of the Debian project estimated that its 230 million lines of source code (230M SLOC) would have required 60,000 person-years of development at a cost of more than $8B USD[13] using proprietary assumptions, but clearly this was achieved at a very small fraction of that estimate.

As of late 2009 there are more than 70[14] licenses that are officially recognized as Open Source Licenses. These licenses govern software that spans the software gamut from operating systems (like Linux) and compilers (like GCC and G++) to web servers and clients (like Apache and Firefox) to leading applications and technologies in bioinformatics, statistical display and analysis, 2-D and 3-D paint, illustration and animation software, databases, content management, customer relationship management, accounting, and geographic information systems. And even these categories encompass but a few of the 10,000+ open source software packages that are freely available for major Linux distributions such as Fedora[15], Ubuntu[16], and Red Hat Enterprise Linux[17].

A study published by SAP Research[18] finds that from 1996-2006, open source software consistently doubled in size (and presumably capability) every 12.5 months, to 1 Billion source lines of code (SLOC). This is astonishing both because using conventional metrics this would have an implied cost to produce of more than $25B USD, and because such a rate of production implies that open source software has achieved what proprietary, notoriously[19], has not: a Moore's Law for software. And this has enormous implications, because exponentially lower production costs means faster and greater return on investment (ROI) and a lowering of barriers to innovation for all economic sectors that *use* ICT.

The documented failure of the industrial model of software development 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 the development of software[20]. Fred Brooks, the IBM executive responsible for the project[21], hired the best coders, managers, analysts—whatever it took. And IBM ended up spending more money to produce that operating system than the US Government spent on the Manhattan Project, a fact that greatly troubled IBM's CEO at the time, Thomas Watson Jr. The product shipped—barely—in 1965, and though the product was a great commercial success, IBM's CEO really wanted to understand how it was possible that the software part of the project was so much more difficult and costly than the hardware part. He put the question to Brooks, but Brooks himself did not have a ready answer. Ten years later Brooks answered the question in the form of a book titled *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 multi-billion dollar software projects like Microsoft's Vista slipped and slipped and slipped again, running wildly over budget and why, when it finally did ship, it did so without the very features that justified the project in the first place. Thus, more thirty years later, we have the Gartner Group (who famously predicted the success of the Personal Computer in enterprise computing) making the only logical prediction that can be inferred from The Mythical Man Month and Microsoft's own experience[22]:

> Vista will be the last major release of Windows in its current form. The current, integrated architecture of Microsoft Windows is unsustainable - for enterprises and for Microsoft. Each new version integrates more functionality and features, adding to the scale and complexity of the operating system.

While IBM and others struggled to industrialize software, the mainstream industrial model itself was beginning to show weakness across the board. In the 1940s, Dr. W. Edwards 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. However, Deming did find a receptive audience in Japan, and the transformation that resulted from his approach has set the standard for modern industry, lowering costs, accelerating innovation, reducing environmental impact, improving safety, and inspiring pride in workmanship[23].

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 a 1987 SEI Technical Report[24]. First adopted not in the US, but offshore, in Chennai, India[25], it was credited with dropping software defect density 98% while doubling profits. Though these first attempts have now been retired[26], the urgent question remains: why is software quality so low[27]?

In 2005, Bill Gates gave a keynote speech at the RSA Security conference where he said "We spend over US$6 billion a year on research and development. I'd say that over a third of that is directly security-focused, and the other two-thirds all tie in and relate to that security work, all the new code being reviewed and going through the threat model, a pretty dramatic thing there"[28]. Why is it 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? 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 place a higher value on control than on innovation or quality or transparency or user participation. It is a failure to understand both the negative lessons that Brooks have taught, as well as the positive lessons that Deming taught. And perhaps it is because those who have achieved great status, wealth, and power are rarely willing to trade all three for a new system that requires accepting new, disruptive assumptions.

The most significant transformation in ICT has been 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[29], and prior to 1990 there had been dozens of attempts to create such systems, the most popular being the French MiniTel system[30]. 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[31].

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[32]:

> [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.

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 (hackers[33]) 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[34]. **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 180,000 projects and has more than 1.9M registered users as of December 2008[35]. Extrapolating from the extensive FLOSS (Free, Libre, and Open Source Software) survey of 2002[36] (and updated in 2005[37]) there were over 490,000 *sourceforge.net* developers in 2006 [when the thesis of this section was first developed—tiemann] who spend more than 10 hours a week or more tending their open source projects[38]—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 (using numbers that were contemporaneous with the FLOSS survey data, October 2006):

1. If all 61,000 employees[39] wrote code, they would have to work over 80 hours/week
2. If Microsoft's $6.6B/year R&D budget[40] 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. Moreover, we find precisely the kind of improvements that Deming would have predicted by taking a transformative approach: according to findings published by Coverity [references below], 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)[41]

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[42]**

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 (nor restricted to just Linux developers)[43]**

2008: Funded by the Department of Homeland Security, the survey was further expanded to scan 250 OSS projects consisting of more than 55 MLOC. **Software defect density had been reduced an additional 16% since 2006, and PHP, which had the highest measured defect density in the LAMP stack in 2006 was one of 11 "perfect" projects, having a measured defect density of zero.[44]**

2009: According to Coverity, the overall integrity, quality, and security of open source software is improving, and open source developers are actively improving [their] software. Rung 2 projects (zero defects at Rung 1 level) have increased more than threefold, from 11 to 36, and open source project involvement has increased more than 50% since 2008.[45]

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[46]. The world's largest public search engine, google.com, is built on open source[47,48,49]. Open source has become a fixture in tech-heavy disciplines such as financial services[50], military intelligence[51], online retailing[52], and next-generation cellular telephones and base stations[53] and handsets, including the recently announced Google Android project/product[54]. And open source is helping to bridge the digital divide for millions of children who are literally off the grid[55]. The model has even informed the strategy of the Bill & Melinda Gates Foundation to better combat infectious diseases[56]:

> [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.

Before explaining how open source can remedy the $1T USD problem, let us dispose of a contrary view, which is that in 2008 Open Source Software *costs* the IT industry $60B per year in lost licensing fees[57]. Put another way, this study found that because of open source software, IT customers spent $60B less than they would have otherwise. Clearly this is a case of glass-half-full/glass-half-empty: the customers who were able to spend less on ICT were able to create more value for themselves[58]. We shall now see that this $60B/year savings is just a small fraction of the total potential savings that open source solutions can deliver on a global basis.

In 2006 Open Source software and services earned $1.8B USD[59] as compared with $235B USD in packaged software sales[60] (which likely pulled through an additional $235B USD in support and services[61]). No matter how one looks at it, Open Source solutions represent less than 1% of global software spend, and yet now enable a reduction of more than 25% of such spending (because $60B is more than 25% of $235B). More impressively, Open Source solutions represent less than 0.1% of global ICT spend, and have already been estimated to deliver back 2% in total returns ($60B is approx 2% of $3T). With these kinds of numbers, the idea of spending half one's budget on open source software and half on proprietary becomes meaningless: **the whole problem could be solved twice over with Open Source solutions for 10% of what is being spent right now**.

At this time, in this moment, the choice is clear: it is time for regime change in the software industry. Open Source software is the key to unlocking 21st century economics and solving 21st century problems. The proprietary software model is unsustainable. It did not scale to meet the challenges of the 1960s, and 50 years later it surely does not scale to address the challenges we now confront. We can no longer afford to pay so much to so few for so little in return. Open Source allows us to invest equally well financial and intellectual capital, and it gives us all the freedom to direct that investment as we each see fit—a scalable model that grows stronger with each additional contributor. The cost savings of Open Source software alone is enough to restore the health of the global economy. When we consider the additional innovation potential that Open Source provides compared to proprietary software, the economic benefits are exponential. Perhaps this explains why an increasing number of local, state, and national governments are adopting formal open source education, development, and procurement policies[62]. And perhaps we should now have hope that these policies, after years of study, are ready for funding and implementation.

1 Hillenius, G., June 26, 2008, EC considers study on migration to Open Source, http://www.osor.eu/news/eu-ec-considers-study-on-migration-to-open-source/?searchterm=policy
2 Buxbaum, P., Navy to focus only on open systems, March 06, 2008, http://www.fcw.com/online/news/151858-1.html
3 Deshpande, A. and Riehle, D., The Total Growth of Open Source, March 14, 2008 http://www.riehle.org/publications/2008/the-total-growth-of-open-source/
4 Mah, P., Gartner: Worldwide IT spending will top $3 trillion, October 10, 2007, http://blogs.techrepublic.com.com/tech-news/?p=1348
5 Atwood, J., The Long, Dismal History of Software Project Failure, May 16, 2006, http://www.codinghorror.com/blog/archives/000588.html
6 Tiemann, M., Software Industry vs. Software Society: Who Wins in 2020, September 2006, http://people.redhat.com/tiemann/STS-Forum-Tiemann-2006.pdf
7 The Standish Group, The CHAOS Report 2009 On IT Project Failure, http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure
8 Sessions, Roger, The Cost of IT Failure, http://simplearchitectures.blogspot.com/2009/09/cost-of-it-failure.html
9 Sessions, Roger, Comment Sept 30 2009, http://simplearchitectures.blogspot.com/2009/09/cost-of-it-failure.html?showComment=1254343601086#c6976121547891430058
10 Tiemann, Michael, http://opensource.org/node/467
11 D'Monte, Leslie, http://www.business-standard.com/india/news/open-source-software-can-save-india-2-bn/369858/
12 Wheeler, D., Counting Source Lines of Code (SLOC), June 2001, http://www.dwheeler.com/sloc/
13 Amor-Iglesias, J., González- Barahona, J., Robles-Martínez, G., Herráiz-Tabernero , I., Measuring Libre Software Using Debian 3.1 (Sarge) as A Case Study: Preliminary Results, http://www.upgrade-cepis.org/issues/2005/3/up6-3Amor.pdf
14 Open Source Initiative, Licenses By Name, http://opensource.org/licenses/alphabetical
15 Fedora Project Home Page, http://fedoraproject.org/
16 Ubuntu Project Home Page, http://www.ubuntu.com/
17 Red Hat Enterprise Linux Product Page, http://www.redhat.com/rhel/
18 The Total Growth of Open Source, http://dirkriehle.com/2008/03/14/the-total-growth-of-open-source/
19 UNCTAD Report, 2003, http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf, p. 95
20 Wikipedia reference, http://en.wikipedia.org/wiki/The_Mythical_Man-Month
21 Frederick Phillips Brooks Biography, Updated June 21, 2006, http://www.cs.unc.edu/~brooks/FPB_BIO.CV.06.2006.pdf
22 Moulds, J., Microsoft's Vista release may be last 'big bang', February 1, 2007, http://www.telegraph.co.uk/finance/2803555/Microsofts-Vista-release-may-be-last-big-bang.html
23 Magnier, M., Rebuilding Japan, October 25, 1999, http://deming.org/index.cfm?content=651
24 Humphrey, W., Sweet, W., A Method for Assessing the Software Engineering Capability of Contractors, http://www.sei.cmu.edu/publications/documents/87.reports/87.tr.023.html
25 Business Week Cover Story, Will Bugs Eat Up the U.S. Lead in Software? December 6, 1999, http://www.businessweek.com/1999/99_49/b3658020.htm
26 http://www.sei.cmu.edu/cmmi/adoption/migration.html (retired link, sorry)
27 Levinson, M., Let's Stop Wasting $78 Billion a Year, Ocober 15, 2001, http://www.cio.com/archive/101501/wasting.html
28 Remarks by Bill Gates, February 15, 2005, http://www.microsoft.com/presspass/exec/billg/speeches/2005/02-15RSA05.aspx
29 Bush, V., As We May Think, July 1945 http://www.theatlantic.com/doc/194507/bush
30 Wikipedia reference, http://en.wikipedia.org/wiki/Minitel
31 Webcraft Web Server Survey, as of January 2009, http://news.netcraft.com/archives/web_server_survey.html
32 Tim Berners-Lee Home Page, http://www.w3.org/People/Berners-Lee/FAQ.html#What2
33 Raymond, E., The Hackers Dictionary, http://www.ccil.org/jargon/jargon_23.html#TAG833
34 Von Hippel, E., Democratizing Innovation, 2005, http://web.mit.edu/evhippel/www/democ1.htm

35 http://alexandria.wiki.sourceforge.net/What+is+SourceForge.net%3F
36 FLOSS FINAL REPORT, June 2002, http://flossproject.org/floss1/stats_5.html
37 FLOSSPOLS Research Home Page, http://www.flosspols.org/research.php
38 FLOSS FINAL REPORT, June 2002, http://flossproject.org/floss1/stats_7_2.html
39 http://finance.yahoo.com/q/pr?s=MSFT
40 http://finance.yahoo.com/q/is?s=MSFT&annual
41Fisher, D., Linux Kernel Review Shows Far Fewer Flaws, December 14, 2002, http://www.eweek.com/c/a/Linux-and-Open-Source/Linux-Kernel-Review-Shows-Far-Fewer-Flaws/
42 Kerner, S., Study: Linux Code Grows as Defects Decline, August 3, 2005, http://www.internetnews.com/dev-news/article.php/3524911
43 Kerner, S., Coverity Study Ranks LAMP Code Quality, March 6, 2006, http://www.internetnews.com/stats/article.php/3589361
44 Coverity Scan Report 2008, http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2008.pdf
45 Coverity Scan Report 2009, http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2009.pdf
46Stein, L., How Perl Saved the Human Genome Project, April 1, 1997, http://www.ddj.com/184410424
47 Vaughn-Nichols, S., Google Opens Up About Open Source, October 26, 2005, http://www.eweek.com/c/a/Linux-and-Open-Source/Google-Opens-Up-About-Open-Source/
48 http://www.informationweek.com/news/software/open_source/showArticle.jhtml?articleID=187202790
49 http://www.informationweek.com/news/software/linux/showArticle.jhtml?articleID=192300293
50 Galli, P., Open Source Making Inroads on Wall Street, April 24, 2006, http://www.eweek.com/c/a/Linux-and-Open-Source/Open-Source-Making-Inroads-on-Wall-Street/
51 http://www.nsa.gov/selinux/
52 O'Reilly, T., Ask Tim, February 2004, http://oreilly.com/pub/a/oreilly/ask_tim/2004/amazon_0204.html
53 LinuxWorld awash with Linux phone buzz, August 16, 2006, http://www.linuxdevices.com/news/NS2577652026.html
54 http://code.google.com/android/
55 http://laptop.org/en/
56 Bollier, D., Is Hell Freezing Over? Bill Gates Embraces the Knowledge Commons, July 21, 2006, http://onthecommons.org/content.php?id=885
57 Rosenberg, D., Study Finds "Free Open Source Software Is Costing Vendors $60 Billion", April 16, 2008, http://news.cnet.com/8301-13846_3-9920202-62.html
58 Tiemann, M., $60B less for proprietary software = $60B more customer value, August 23, 2008, http://www.opensource.org/node/364
59 Lawton, M., Notarfonzo, R. Worldwide Open Source Software Business Models 2007–2011 Forecast: A Preliminary View. IDC Inc.
60 Software & Information Industry Association. Packaged Software Industry Revenue and Growth, 2006. Available from http://siia.net/software/
61 Author's estimate based on Standish Group and other studies
62 Lewis, J., Government Open Source Policies , August 2007 , Center for Strategic and International Studies http://www.csis.org/media/csis/pubs/070820_open_source_policies.pdf