Estimating the Total Development Cost of a Linux Distribution

By Amanda McPherson, Brian Proffitt, and Ron Hale-Evans

OCTOBER 2008

Estimating the Total Development Cost of a Linux Distribution

By Amanda McPherson, Brian Proffitt, and Ron Hale-Evans

The Linux operating system is the most successful open source project in history, but just how much is the software in a Linux distribution "worth"? In 2002, David A. Wheeler published a well-regarded study that examined the Software Lines of Code present in a typical Linux distribution. His findings? The total development cost represented in a typical Linux distribution was $1.2 billion. We've used his tools and method to update these findings. Using the same tools, we estimate that it would take approximately $10.8 billion to build the Fedora 9 distribution in today's dollars, with today's software development costs. Additionally, it would take $1.4 billion to develop the Linux kernel alone. This paper outlines our technique and highlights the latest costs of developing Linux.

The Linux operating system is the most popular open source operating system in computing today, representing a $25 billion ecosystem in 2008.[1] Since its inception in 1991, it has grown to become a force in computing, powering everything from the New York Stock Exchange to mobile phones to supercomputers to consumer devices.

As an open operating system, Linux is developed collaboratively, meaning no one company is solely responsible for its development or ongoing support. Companies participating in the Linux economy share research and development costs with their partners and competitors. This spreading of development burden amongst individuals and companies has resulted in a large and efficient ecosystem and unheralded software innovation.

Over 1,000 developers, from at least 100 different companies, contribute to every kernel release. In the past two years alone, over 3,200 developers from 200 companies have contributed to the kernel.[2] It's important to note that the kernel is just one small piece of a Linux distribution. A distribution is actually made up of multiple components including the kernel, the GNOME and KDE desktop environments, the GNU components, the X window system, and many more. The total of individual developers contributing to these projects surely numbers in the thousands.

1 IDC "The Role of Linux Commercial Servers and Workloads", 2008

2 Greg Kroah-Hartman, Jonathan Corbet, and Amanda McPherson, "Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It", April 2008, http://www.linuxfoundation.org/publications/linuxkerneldevelopment.php

Because Linux has been developed collectively, there is no single source for cost estimates of how much it has taken to develop the technology. In 2002, David A. Wheeler published a well-regarded study that examined the Software Lines of Code present in a typical Linux distribution (Red Hat Linux 7.1)[3]. He concluded—as we did—that Software Lines of Code is the most practical method to determine open source software value since it focuses on the end result and not on per-company or per-developer estimates.[4] Using the industry standard tools he developed to count and analyze SLOC, he determined that it would cost over $1.2 billion to develop a Linux distribution by conventional proprietary means in the U.S.

That was six years ago. Since Linux is innovating and growing at a high rate year over year, the $1.2 billion number needed to be updated to reflect the true value of development reflected in Linux today (and the rising cost of software development itself). For this paper, the Linux Foundation set out to determine the total development cost represented in a typical Linux distribution and to update that $1.2 billion number widely used since its publication in 2002.

We analyzed the Fedora 9 distribution, which was released on May 13, 2008. It is a popular and well-used distribution and is the base for Red Hat Enterprise Linux which represents a large percentage of the Linux market. It is also a direct descendant of the Red Hat Linux 7.1 software analyzed by Wheeler in his original paper.

For this study, we used David A. Wheeler's well-known SLOC tool, SLOCCount. SLOCCount makes use of the industry standard **CO**nstructive **CO**st **MO**del (COCOMO), an algorithmic Software Cost Estimation Model[5] developed by Barry Boehm[6]. The model uses a basic regression[7] formula, with parameters that are derived from historical project data and current project characteristics.[8] We updated his study from 2002 to include the growing code base of the Linux kernel and other packages as well as the higher annual salary of a software developer. (More detail on this follows in the Approach section of the paper.)

Using this approach, we estimate that it would take $10.8 billion to develop the Linux distribution Fedora 9 by traditional proprietary means in year 2008 dollars.

3 David A. Wheeler, "More Than a Gigabuck: Estimating GNU/Linux's Size" 2001 (revised 2002)

4 http://en.wikipedia.org/wiki/Source_lines_of_code

5 http://en.wikipedia.org/wiki/Estimation_in_software_engineering

6 http://en.wikipedia.org/wiki/Barry_Boehm

7 http://en.wikipedia.org/wiki/Regression_analysis

8 http://en.wikipedia.org/wiki/COCOMO

Approach

Our basic approach was to:

1. **Install the source code files in uncompressed format**; this requires downloading the source code and installing it properly on our test machines.
2. **Count the number of source lines of code (SLOC)**; which requires a careful definition of SLOC.
3. **Use an estimation model (based on COCOMO practices)** to estimate the effort and cost of developing the same system in a proprietary manner.

For those who are interested in how we installed and analyzed the source code, please see the Appendix.

To update the 2002 study, it was decided that we would use the same base code used in the 2002 study, Fedora—the community distribution that forms the basis of Red Hat Enterprise Linux. For our examination, we decided to use Fedora 9. We counted all the available Fedora 9 packages that were made public in the mirrors.kernel.org mirror archive. It was decided to use all of the packages because we did not wish to use an arbitrary definition of which flavor of Fedora 9 we would measure. Fedora contains significantly more software than Red Hat's enterprise version. One reason for this is a diverse community is involved in building Fedora, not just a company. Using the SLOCCount application is a remarkably simple task: it is simply a matter of pointing it at the correct directory where the source code resides, and letting it run. A detailed explanation of how the program works and how to use it is still available at Wheeler's website[9].

To calculate the costs for these distributions, a base salary was found for computer programmers from the US Bureau of Labor Statistics. According to the BLS, the average salary for a US programmer in July, 2008 was $75,662.08[10]. This was the salary amount used in our SLOCCount run. (Of course most software development these days is global so using a US-only salary number is somewhat specious. One avenue of continued exploration would be to determine a global average salary to use as a baseline for production costs.)

It should be noted that salary alone is not the sole determinant of software development costs. In his articles, Wheeler notes the presence of the overhead parameter within SLOCCount. This takes into account the costs of testing, equipment, company operating costs, and total compensation for the developer. This parameter is also known as the wrap rate.

As just one example of how these wrap rate costs can quickly increase, in the United States, the average compensation percentage (sick days, vacation, insurance benefits) for a professional employee is 29.0 percent[11]. Thus, the programmer making the average US salary figure of $75,662.08 is actually costing the employer $97,604.08 in compensation alone. This is just one piece of the total wrap rate pie.

Wheeler's own estimate of the total wrap rate value is 2.4. Though this figure clearly varies by company and region, further research did not show any significant evidence that this figure needed adjustment for our tests.

9 http://www.dwheeler.com/sloccount/sloccount.html

10 Bureau of Labor Statistics, CES Database http://www.bls.gov/ces/#data

11 Bureau of Labor Statistics, http://www.bls.gov/news.release/ecec.t05.htm (March 2008)

Source Code and Estimated Cost Results

Finally, given all the assumptions shown previously, the SLOC and estimated production values for Fedora 9 are as follows.

Total Physical Source Lines of Code (SLOC)	**204,500,946**
Development Effort Estimate, Person-Years (Person-Months) (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))	59389.53 (712674.36)
Schedule Estimate, Years (Months) (Basic COCOMO model, Months = 2.5 * (person-months**0.38))	24.64 (295.68)
Total Estimated Cost to Develop (average salary = $75,662.08/year, overhead = 2.40).	**$10,784,484,309**

As an aside: the original SLOCCount annual salary figure was $56,286 in year 2000 dollars, so it's of interest to see just how today's costs translate to those same 2000 costs. To find this, we multiplied the total cost to develop Fedora 9 in 2008 dollars ($10,784,484,309) by 0.744, which is the ratio between the year 2000 programmer's salary normally used by SLOCCount ($56,286) and the July 2008 salary we found ($75,662.08). The result is just over $8 billion to develop Fedora 9 in year 2000 costs. This gives the reader a good indication of just how much code and functionality has been added to a Linux distribution in these six years.

Focusing on the Linux Kernel and the Intermediate Model

As research was conducted for this paper, it quickly became apparent that not all of the components within a given Linux distribution needed to be treated equally. Some projects, by their very nature and complexity, need different consideration under the COCOMO model.

Nowhere was this more apparent than in Wheeler's own follow-up to his 2002 article, where he highlighted the need for a different estimate model for the Linux kernel itself.[12] In this new article, Wheeler maintained that because the Linux kernel code is typically more complex than an "average" application—among other things—it requires an analysis that goes beyond the basic COCOMO model. A user space application like Mozilla, for instance, is much easier to code line by line since it's abstracted at a much higher level and has to handle far less tasks. A modern and enterprise-class operating system kernel is asked to do a great number of extremely complex things, all at once.

Wheeler explains that instead of using the standard method used for analyzing all elements of a distribution, one should use the intermediate COCOMO model for counting just the kernel. The intermediate method has more variables to choose from, and therefore should be more accurate for analyzing a complex piece of software on its own. In the 2004 article, he details his estimates for these parameters, which have the net effect of adjusting the overall –effort factor value in SLOCCount from the default of 3 to a new value of 4.64607. At the same time, the –effort exponent value is also changed to 1.12, which is the value assigned to semi-detached software projects under the intermediate COCOMO software estimation model.

With these new parameters in mind, a separate test was run on the stock Linux kernel that was included within Fedora 9, linux-2.6.25.i686:

Total Physical Source Lines of Code (SLOC)	**6,772,902**
Development Effort Estimate, Person-Years (Person-Months) (effort model Person-Months = 4.64607 * (KSLOC**1.12))	7557.4 (90688.77)
Schedule Estimate, Years (Months) (Basic COCOMO model, Months = 2.5 * (person-months**0.38))	15.95 (191.34)
Estimated Average Number of Developers (Effort/Schedule)	473.96
Total Estimated Cost to Develop (average salary = $75,662.08/year, overhead = 2.40).	**$1,372,340,206**

12 David A. Wheeler, "Linux Kernel 2.6: It's Worth More!" 2004 (revised 2007) http://www.dwheeler.com/essays/linux-kernel-cost.html

Limitations and Advantages to this Study's Approach

There is no perfect way to estimate the value of something as complex and evolving as Linux. While this method we feel is the best approach, it likely over-counts some aspects of value, while under-estimates others. Here is a summary of those limitations:

- **The COCOMO model was designed from research on proprietary software development.** Because of that, we feel it undercounts the testing complexity inherent in open source, collaboratively developed software projects like Linux. Because of the massive amounts of change, both small and large, and the distributed nature of developers and stand-alone projects, the testing burden on Linux ecosystem participants is at an order of magnitude higher than for proprietary, stand-alone companies.
- **Another difficultly in measuring the value of a Linux distribution is simply determining what comprises a Linux distribution to begin with.** We included all packages on Fedora's public source-code repository. But there are certainly smaller distributions of even Fedora (the LiveCD version for instance). There are also distributions that are larger; Debian GNU/Linux's[13] repository is bigger than Fedora's, for example. There is also a large amount of open source software that isn't found in either distribution. It's a big open source universe out there, so we tried to rely on one distribution's available packages.
- **The biggest weakness in SLOC analysis is its focus on net additions to software projects.** Anyone who is familiar with kernel development, for instance, realizes that the highest man-power cost in its development is when code is deleted and modified. The amount of effort that goes into deleting and changing code, not just adding to it, is not reflected in the values associated with this estimate. Because in a collaborative development model, code is developed and then changed and deleted, the true value is far greater than the existing code base. Just think about the process: when a few lines of code are added to the kernel, for instance, many more have to be modified to be compatible with that change. The work that goes into understanding the dependencies and outcomes and then changing that code is not well represented in this study.
- **Collaborative development means you'll often have multiple individuals or groups working on different approaches to solving the same technical problem with only one of those approaches "winning" inclusion in the final version.** Since the "losing" approaches are not committed to the final shipping product, this SLOC approach does not take into account the development effort for those approaches.
- **There are significant software lines of code differences between distributions and even different versions of distributions, so it's misleading to consider one analysis to be the canonical estimate of the value of "Linux."** There are many different options in what to analyze and we chose only one. For that reason, we find the estimates of discrete packages that all distributions share (the kernel for instance) to be more interesting.

13 For more information on Debian source-code counts, see Counting Potatoes: The size of Debian 2.2 (http://people.debian.org/~jgb/debian-counting) and 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)

- **Unfortunately this method equates value to quantity.** The Linux community takes "bloat" very seriously, but at the same time Linux contains a lot of code—like old drivers—not used very often. Including this code is important though, since both architecture-specific code as well as drivers are included in Linux (unlike other operating systems). As an effect of this, this number would be larger than for other operating systems that do not include these components within the OS itself.
- **This study assumes all development would take place in the US, with the associated cost of US labor.** Most software development is global in nature and its labor costs would vary accordingly.
- **The numbers reflected in this study represent how much it would cost to develop the software in a Linux distribution today, from scratch.** It's important to note that this estimates the cost but not the value to the greater ecosystem since changing such a core piece of technology so widely used would have huge and far-reaching economic impact.

How Has Linux Grown?

This study also does not take into account the yearly development costs to incrementally update and grow the Linux code base. Based on these numbers (or anyone's direct experience with a Linux distribution) it's easy to see how the functionality included in a Linux distribution has exploded over the last six years.

For instance, Fedora Linux 9 includes just over 204 million physical source lines of code (SLOC), compared to 30 million lines in Red Hat Linux 7.1 (issued in 2002), and over 17 million lines in version 6.2. So in just six years, the code base grew by 174 million lines of code. Using the COCOMO cost model, we estimate Fedora 9 to have required about 60,000 person-years of development time (as compared to 8,000 person-years for Red Hat 7.1 and 4,500 person-years for version 6.2). Thus, Fedora 9 represents a roughly 680% increase in size, a 750% increase in effort, and a 900% increase in traditional development costs over Red Hat Linux 7.1. This is due to an increased number of mature and maturing open source/free software programs available worldwide. Such growth shows that Linux has a great deal of momentum: the continual addition of open source packages strengthens the applications set available to Linux users and in turn makes Linux that much more attractive as a computing platform.

The modular nature of Linux (in its composition of a distribution) is also apparent by scanning the enclosed list of top ten packages included in a distribution. Given the time and interest, it would be interesting, for instance, to analyze the top components of an embedded Linux distribution and the development costs associated with that segment of computing.

The impact in Linux innovation is not just measured in added lines. As the recent "Who Writes the Linux Kernel" paper states: "Over these releases, the kernel team has a very constant growth rate of about 10% per year, a very impressive number given the size of the code tree. But the kernel is not just growing. With every change that is made to the kernel source tree, lines are added, modified, and deleted in order to accomplish the needed changes."[14]

While the Linux kernel probably has a higher change rate than most other components of Linux, our numbers reflect that as a whole Linux is growing and changing every year at a steady clip. Since the Linux kernel is just one small (but very important) component of a Linux distribution, the iterative development costs poured into Linux per year are substantial but not examined in this study.

14 Greg Kroah-Hartman, Jonathan Corbet, and Amanda McPherson, "Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It", April 2008, http://www.linuxfoundation.org/publications/linuxkerneldevelopment.php

Conclusions

So do we now know what Linux is "worth?" While it may not be a question capable of being answered completely, some things are very clear: The true value of Linux springs from the ability to re-use it and the tremendous flexibility that it creates.

Just imagine a computing world where Linus Torvalds didn't allow (in fact force) users of Linux to allow others to re-use their contributions. Would there be a Google if they didn't have the free use of Linux and the ability to modify it to suit their needs? Could there be the expanding new category of sub-$350 ultra mobile PCs without the free use of a $10.8 billion piece of software? Would Amazon be able to build its new line of Kindle reading devices without a free piece of $1.4 billion R&D to power it? More than just money, the software in a Linux distribution represents time. The economics in each of these examples would not have been possible had these companies been forced to pay per-device or per-server license fees to any one company or had to devote the thousands of person-years of development time to create this software.

What can we learn from this study? The substantial development costs represented in a community-based Linux distribution reflect its increasing value and importance in the world of computing. The companies and individuals who work on Linux-related projects and build this value profit by sharing the development burden with their peers (and sometimes competitors.) Increasingly it's becoming clear that shouldering this research and development burden individually, as Microsoft has done, is an expensive approach to building software. While monopoly position in the past has allowed them to fund this massive development, we believe that in the future competition from collaborative forces will make such an isolated position untenable.

As we can see from this study and through the explosion of Linux throughout all areas of computing, collaborative development creates enormous economic value. Companies such as IBM, Intel, HP, Fujitsu, NEC, Hitachi, Google, Novell, Oracle, Red Hat, and many others have all participated and profited in the tremendous ecosystem created by this open model of software development. But make no mistake: while companies are participating, individuals are just as important in the expansion of this software and its value. That's how it all began after all.

The data in this paper was generated by *SLOCCount, Copyright (C) 2001-2004 David A. Wheeler*
SLOCCount is Open Source Software/Free Software, licensed under the GNU GPL.
SLOCCount comes with ABSOLUTELY NO WARRANTY, and you are welcome to redistribute it under certain conditions as specified by the GNU GPL license; see the documentation for details.

Acknowledgements

We would like to thank the following people for reviewing and providing feedback on this paper: James Bottomley, Jon Corbet, and David A. Wheeler.

Appendix

For Fedora 9, the installation CD did not contain the source code, so the code was downloaded from the mirrors.kernel.org web site. Because determining what files to analyze would have entered another level of subjectivity into the process, it was decided that all 5547 available open source packages (src.rpms) would be downloaded and installed on the test machine.

After downloading the src.rpms from the FTP server, it was then time to install them. Even though none of the rpms would be installed as binary RPMs (thus potentially being actually installed as applications on the test machine), a procedure was put in place to allow the source RPMs to be built and prepped without root access, modifying a procedure found on an archived Red Hat howto page[15].

The source code was downloaded to the test user's home directory.

Using gedit, a script was then created and saved as .rpmmacros in the same home directory. A destination directory set was created and then the src.rpm files were installed using this command:

rpm -ivh *.src.rpm

After a good amount of time, all 5547 packages were installed, with the specification files (.spec) residing in the rpm/SPECS directory and the source files and graphics populating the rpm/SOURCES directory.

At this stage, the src.rpm files would have to built and prepped, which would align all of the applications' source code into their own directories in the rpm/BUILD directory. To do this, the following command was used:

rpmbuild -bp --nodeps *.spec

Once this command was run, all of the packages were installed properly within the BUILD directory.

The actual counting could then begin. Because a distribution is not a single software project, it should not be counted as such. SLOCCount provides a parameter to compensate for this: --multiproject.

For the Fedora 9 report, the command used was:

sloccount --multiproject --personcost 75662.08 /usr/src/rpm/BUILD/ &> sloc.txt

15 http://www.redhat.com/archives/rpm-list/2002-February/msg00030.html

For further examination, following are the top 10 packages from the Fedora 9 source code:

SLOC	Directory	SLOC-by-Language (Sorted)
5961705	kernel-2.6.25i686	ansic=5727336, asm=216356,perl=6019, cpp=3962, yacc=2901, lex=1824, objc=613,python=331,lisp=218, pascal=116, awk=96
4991916	OpenOffice.org	cpp=4518218, java=262028, ansic=109607, perl=66501, sh=11288, yacc=6657, cs=6600, python=3023, lex=2474, asm=2453, lisp=920, awk=734, objc=594, pascal=407, csh=301, php=104, sed=7
3118105	Gcc-4.3.0-2 0080428	ansic=1559056, java=646496, ada=478683, cpp=305899, f90=50636, asm=25272, sh=19070, exp=11829, fortran=7651, objc=3539, perl=2732, ml=2485, yacc=1440, pascal=1092, awk=764, python=582, lex=461, tcl=381, haskell=37
2664636	Enterprise Security Client 1.0.1	cpp=1725793, ansic=862640, perl=27244, asm=16749, sh=16435, cs=6232, java=5325, python=3077, lex=306, php=244, pascal=230, csh=132, objc=97, yacc=79, ada=49, sed=4
2216353	eclipse-3.3.2	java=2089544, ansic=96269, cpp=24302, jsp=3965, perl=1325, sh=878, python=46, php=24
2123390	Mono-1.9.1	cs=1862734, ansic=257228, sh=1998, perl=807, asm=335, yacc=288
2088834	firefox-3.0	cpp=1280521, ansic=743238, asm=32703, sh=14686, perl=7177, python=6549, java=2668, makefile=2451, objc=867, lisp=256, pascal=159, awk=10
1792482	bigloo3.0b	ansic=1621378, lisp=143716, sh=10296, java=8233, cs=7846, cpp=849, asm=159, yacc=5
1788219	gcc-3.4.6-20060404	ansic=858843, ada=485364, cpp=196337, java=175563, asm=24403, sh=19731, yacc=14038, exp=5657, objc=4408, fortran=2032, perl=898, lex=587, awk=189, pascal=86, haskell=66, sed=17
1543156	ParaView3.2.1	cpp=960400, ansic=509436, tcl=45787, python=19574, perl=3112, yacc=1787, java=1517, sh=644, asm=471, lex=400, objc=28