The Evolution of FreeBSD and Linux

Clemente Izurieta
Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA
01-970-481-6172
cizuriet@colostate.edu

James Bieman
Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA
01-970-491-7096
bieman@colostate.edu

ABSTRACT

Is the nature of Open Source Software (OSS) evolution fundamentally different from that of the traditional and commercially available software systems? Lehman and others conducted a series of empirical studies that found that traditional systems grow at a linear or sub-linear rate. A prior case study of the Linux OSS system suggests that OSS may evolve in a unique manner. Godfrey and Tu found that some aspects of Linux are growing at a super-linear rate rather than a sub-linear rate. Additional studies are necessary before drawing conclusions. Thus, we examine the evolution of FreeBSD and re-analyze the evolution of Linux, and find evidence that the growth of both systems has a linear upper bound, and thus appear to grow at similar rates to that of commercial systems. These results do not support the hypothesis that OSS systems grow at rates that exceed that of traditional systems.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *software evolution.*

General Terms

Measurement, Design, Experimentation.

Keywords

Software Engineering, Evolution, Open Source Software, FreeBSD, Linux, Replication Study.

1. INTRODUCTION

Open source software (OSS) development processes appear to be strikingly different from that of traditional systems. Studies of OSS products and processes can identify these differences, if they exist. Data and source code are readily available. Examples of successful OSS products include Linux, FreeBSD, gcc, Mozilla, Apache, etc.

Although there is controversy concerning the testability of Lehman's laws [8] of software evolution [16], Lehman and his collaborators identified common trends in the growth of commercial systems. Lehman's empirical studies were carried out on numerous E-type commercial systems of different sizes and developed by different organizations. An E-type system is described by Scacchi [16] as *"An application embedded in its setting of use"*. In other words, there is feedback provided by the continuing evolution of the software.

In some of his earliest empirical work [10], Lehman and Ramil showed clear linear growth of the OS/360 operating system from IBM over a number of releases. Further, Lehman and Ramil established clear commonalities in the evolution of software through a series of empirical studies in the *Feedback, Evolution And Software Technology* (FEAST) [11] projects that showed sub linear growth and inverse squared trends among these commercial systems. Lehman and Ramil state *"An example of such similarities is provided by the fact that five of the six FEAST systems display a positive, but predominantly declining, long term, growth rate over releases."* The FEAST project analyzed empirical data from six commercial systems from very different industries, namely, a financial transaction system, an information system, an OS kernel, two real time systems, and a military system.

Even thought the newer studies by Lehman and Ramil are exploratory in nature, they do suggest that feedback driven mechanisms are in place that force an initial linear and long term sub-linear trend in software evolution of commercial systems. We further claim these studies carry over to that of Open Source Software systems.

Lawrence also carried out empirical studies of the software evolution of seven commercial systems [7]. Four are operating systems─ IBM's DOS and OS/360, ICL's VME, and UNIVAC's Omega, one system was a financial application, and the final two were batch systems for inventory control. Lawrence found that both of IBM's operating systems exhibited higher growth rates than the other systems, but the growth rate was linear. The other five systems showed declining growth as the number of releases increased.

Godfrey and Tu [4] studied the evolution of Linux, a well known OSS system. In their study, they *"expected to find that Linux was growing more slowly as it got bigger and more complex."* They expected OSS systems to exhibit a growth rate similar to their industrial counterparts, namely, one approximating an inverse square function [3]. They found evidence suggesting that Linux is growing at a super-linear rate, which contradicts the empirical studies of commercial systems.

The case study carried out by Godfrey and Tu is just one data point, but one that was carried out on a well known OSS system. Although Godfrey and Tu found evidence to suggest super-linear growth in the driver sub-system of the Linux OS, they have also found evidence of linear growth in systems such as FetchMail, X-

Windows, and gcc, and report on evidence of sub-linear growth in Pine (e-mail client) [5, 6].

The Linux project is a *pure* OSS project. The term "pure" is used by Mockus et al. [14] to describe projects without any significant commercial involvement. We thus carried out a case study of FreeBSD [2], another OSS system in the same application domain as Linux. We also re-examined the evolution of Linux, focusing on the growth rates of stable releases. Additionally, we break up the system into its sub-systems and study their growth rates individually as suggested by Gall et al. [3].

Because FreeBSD does not support as many devices as Linux, where devices constitute as much as 60% of the total system size, and because FreeBSD goes through a more demanding testing and acceptance process [1], we expected that this OSS system will more closely adhere to showing evidence of sub-linear growth. We also expected that Linux and its extensive driver subsystem will show a steeper growth rate.

To evaluate our data we will carefully analyze the measures that we obtain from the FreeBSD and Linux project repositories and databases over a period of time and across various releases of *stable* source trees.

Gall et al. [3] suggests dividing the system into subsystems in order to understand the evolution of the sub-parts, otherwise valuable information may be lost. Godfrey and Tu [6] did this by using the file system of Linux to inherently divide the system architecture. This process suppresses any biases imposed by the authors on the subdivision of systems. We also allow the file system structure of both operating systems to serve as this natural sub-division heuristic.

In this paper, we examine the Godfrey and Tu [4] result that Linux has grown at a super-linear rate and see if FreeBSD, another similar OSS system, grows in a consistent manner. Since prior studies found that traditional systems grow at sub-linear rates, we are essentially examining the hypothesis that OSS systems, or at least OSS operating systems, grow at a greater rate than that of traditional systems. Thus, our working null hypothesis is that the growth rates of OSS systems do not differ from the sub-linear rates found in traditional software development. We see if these case studies provide evidence to refute the null hypothesis.

2. THE LINUX AND FREEBSD OPERATING SYSTEMS

The FreeBSD project web site [2] describes this OSS system as an operating system for x86 compatible (including Pentium® and Athlon™), amd64 compatible (including Opteron™, Athlon 64, and EM64T), Alpha/AXP, IA-64, PC-98 and UltraSPARC® architectures. It is derived from BSD, the version of UNIX® developed at the University of California, Berkeley. Similarly, Linux is supported on various platforms including PowerPC, UltraSPARC® architectures, IA-64, etc. Linux was originally developed by Linus Torvalds, but has grown in size considerably thanks to the support of hundreds of contributors.

FreeBSD and Linux are large and successful OSS system projects that depend on the contributions of its developer/user community to continue to evolve. The *community* of the FreeBSD project is made up of a group of *core developers* (7-9), a group of *committers*; which is a trusted developer community, and where committers may be nominated to become part of the core team [1]. Linux follows a similar setup where the vast majority of contributors are volunteers. These communities are distributed geographically and culturally.

FreeBSD uses a CVS (Concurrent Version Control) repository to store information about the code and its history. It is common for OSS projects to use CVS for source code management. FreeBSD and Linux maintain *stable* branches of their source code. Stable code is tested and not experimental. Both operating systems also maintain branches that house the latest developments and are experimental in nature. Linux refers to these experimental branches as *development* branches, whereas FreeBSD calls them *current* branches. We concentrate our efforts on measuring stable releases for both operating systems, as these tend to exhibit more analogous properties to commercial releases; which are what Lehman's empirical studies are based on.

Similar to Godfrey and Tu, we maintain the inherent sub-directory structure of the operating systems as the natural sub-division of the sub-systems. Additional information regarding the various directories and their corresponding descriptions can be found in the OSS systems websites [2, 13].

3. METHODOLOGY

We measure the various stable FreeBSD and Linux releases from their inception. Each release is subdivided further into their file system structure. We capture various metrics, including LOCs (Lines of Code), number of directories, total size in Kbytes, average and median LOC for header (dot-h) and source (dot-c) files, and number of modules (files) for each sub-system and for the system as a whole.

Before each release, commercial systems are thoroughly tested in order to satisfy customers, and are by no means experimental. Stable releases are not part of just a single source code branch, rather, as the software matures, new stable source code branches fork and merge, thus discontinuing previous version trees. Figure 1 shows the evolution of FreeBSD [6] for example, and the Unix website [18] has a similar picture for Linux. Missing releases are not an indication of discontinuities in the source branches; rather the data necessary for our study was not readily available. All FreeBSD releases prior to and including 3.3 were only supported on Intel's 386 architecture. Starting with release 3.4 and the entire 4.x stable branch, FreeBSD was also supported on Alpha, and finally, the 5.x branch of the tree is currently supporting a number of different architectures as shown in the text box next to the source tree. Releases are shown in chronological order from top to bottom. Our study focuses on the releases that are shaded in grey. In total, we study 34 stable releases of FreeBSD and 127 stable releases of Linux. Although this paper does not analyze evolution across systems (various branches), Nakakoji et al [15] studied how code branches merge and split, giving a different view of evolution across systems.

Like Godfrey and Tu's study we use Unix *"wc –l"* to count the LOC in every source file, where a source file is either a *.h or *.c file. The entire system size will be counted as an aggregate of the sub-systems in a specific release. We use a shell script to compute and gather the statistics.

Godfrey and Tu measure size using uncommented LOC rather than source files because "using number of source files would mean losing some of the full story of the evolution of Linux, especially at the subsystem level." Our case study measures system growth using various methods, but we include a count of source files to see if any irregular correlation appears. Our results show growth rates of the various sub-systems and system as a whole by using various measures. We use release dates, release numbers (RNs) as done by Lehman, and cumulative release numbers. We also plot our range against cumulative growth as a percentage of the size of the first release, and as the total size measured in Kbytes. Finally, we provide additional graphs that show the average and median LOC for dot-h and dot-c files.

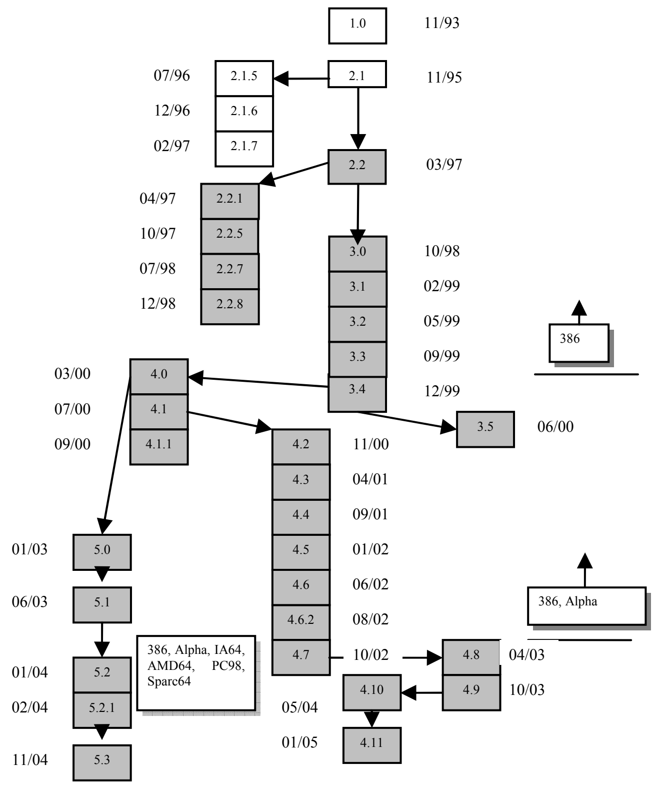

Figure 1. FreeBSD Release Tree.

The case study was carried out as follows:

1. First we generate the source version tree of the various releases of Linux and FreeBSD and the dates when they were released. When plotting curves this would give us additional insights where *jumps* are observed.
2. Download the entire FreeBSD CVS tree. The CVS repository requires approximately 2.6GBytes. Unfortunately the releases are only available starting with version 2.0. For Linux, we downloaded each release individually from the Linux web-site [13].
3. For FreeBSD, we use CVS commands to check out entire releases based on the tagging mechanism. Once the release is checked out, we use various UNIX commands or scripts to generate the desired measures. The same UNIX commands and scripts are used to gather statistics from Linux.
4. Measurements are taken at the system and subsystem levels. We maintain the inherent directory structures of both operating systems to subdivide each system, akin to the methodology that Godfrey and Tu use to subdivide Linux in their case study [5].
5. We plot various data using Microsoft Excel spreadsheets.
6. We analyze the collected information to determine if either or both operating systems show evidence of linear growth. We also determine if these systems are consistent with the observations made by Turski [17] that the development of software follows an inverse square growth function.

4. OBSERVATIONS ON THE EVOLUTION OF FREEBSD AND LINUX

We view and analyze our results from multiple perspectives. We took measurements of the system as a whole, as well as of the individual subsystems.

4.1 Observations at the System Level

Godfrey and Tu plotted growth rates against calendar dates rather than release numbers. Further, Godfrey and Tu suggest that plotting according to release numbers would have led to dips in the function curves because development and stable releases follow different behaviors.

Such dips will occur when plotting releases sequentially; however, since we are only analyzing stable releases, it makes sense to plot them against release numbers. For the sake of completeness, we have also plotted growth rates against release dates; however, due to space limitations, these plots are not included. We have also plotted the cumulative release numbers versus the corresponding cumulative growth of the system measured as a percentage of the original (or first) release size. The plot is similar to the graphs given by Scacchi [16] (page 9) where he plots the evolution of OS/360, Logica FW, Lucent Sys1, and the ICE VME Kernel systems described in various Lehman et al. references.

Figure 2 shows two plots depicting the growth of FreeBSD, while Figure 3 shows these plots for Linux. The sharp spikes in the growth curves are attributed to the expected growth deltas between sequential releases. This is most evident in the growth plots of Linux on Figure 3. For example, we can clearly observe sharp size release increases between source branches 2.0.x and 2.2.0, similarly between branches 2.2.x and 2.4.1, etc. This can be attributed to the increasing popularity of the system and the support for new modules and drivers.

Clearly the curves show growth rates that are at most linear. We also plot each of the different stable branches of evolution against calendar times in parallel because various branches are really developed and released concurrently. After further studying these graphs, we see no evidence to suggest anything other than linear growth for each individual branch. In fact FreeBSD release branch 2.x has decreased in size, while FreeBSD release branches 3.x and 4.x and Linux branches 2.2.x and 2.4.x appear to show an inverse growth property as suggested by Turski [17].

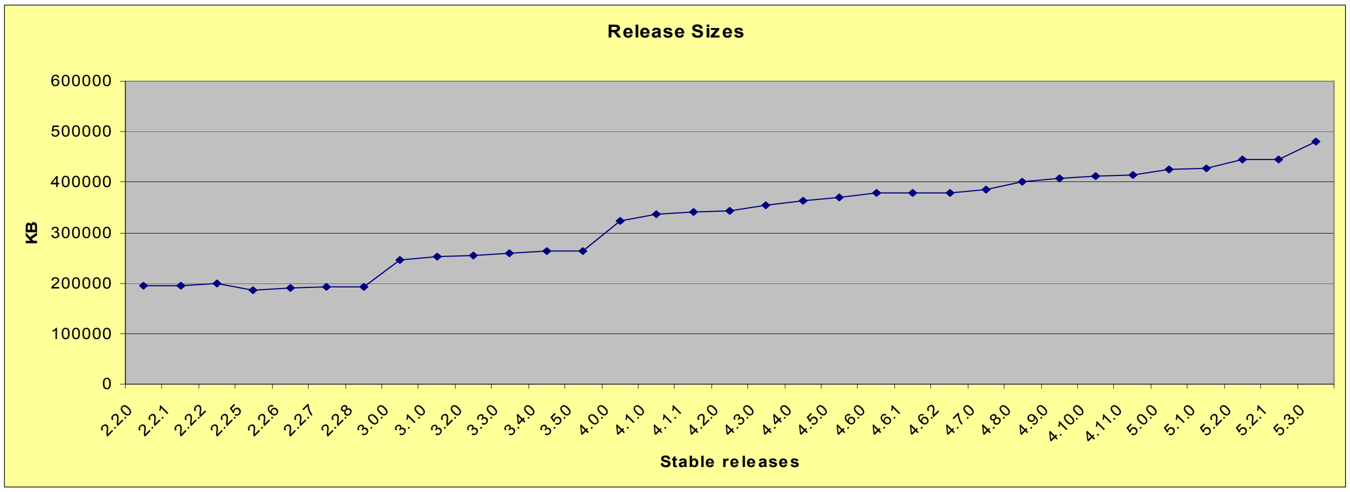

Figure 2A. FreeBSD stable release growth by release number

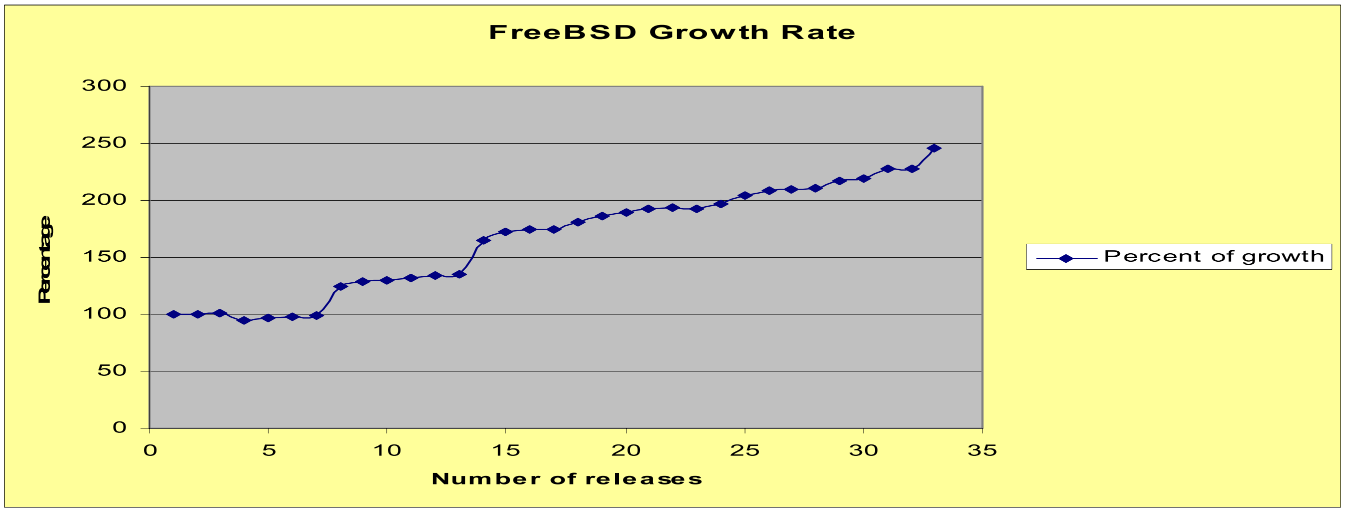

Figure 2B. FreeBSD cumulative growth rate

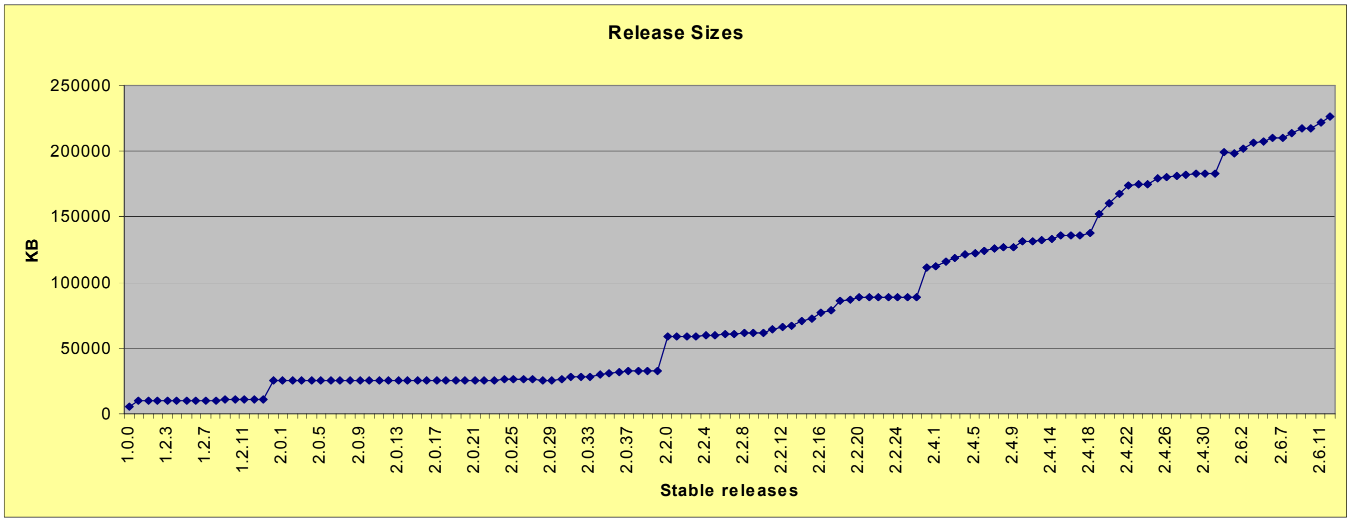

Figure 3A. Linux stable release growth by release number

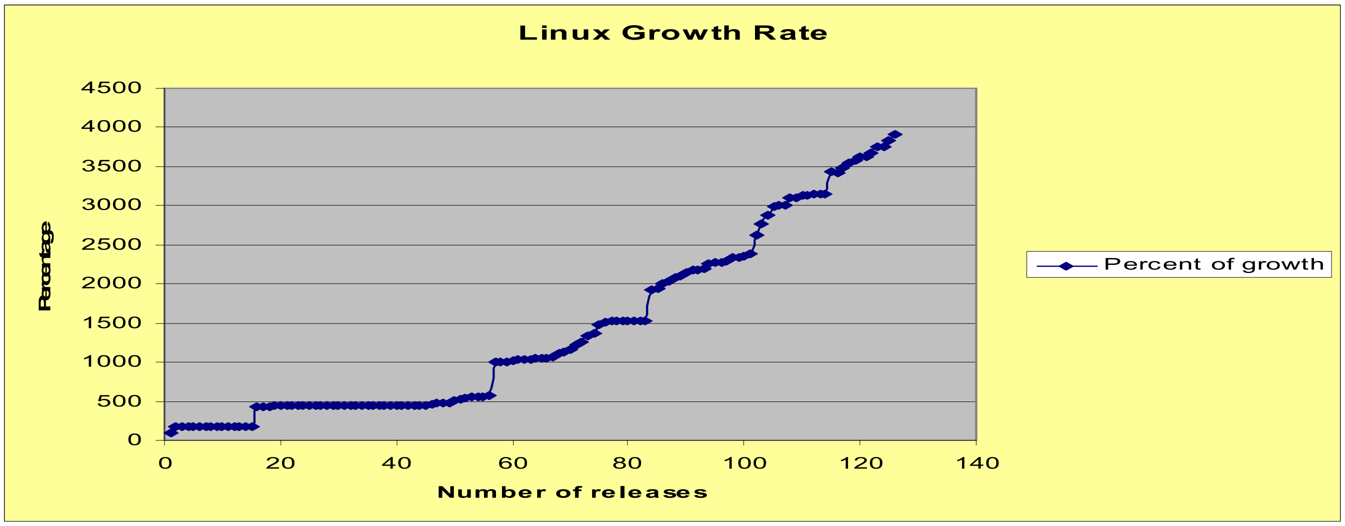

Figure 3B. Linux cumulative growth rate

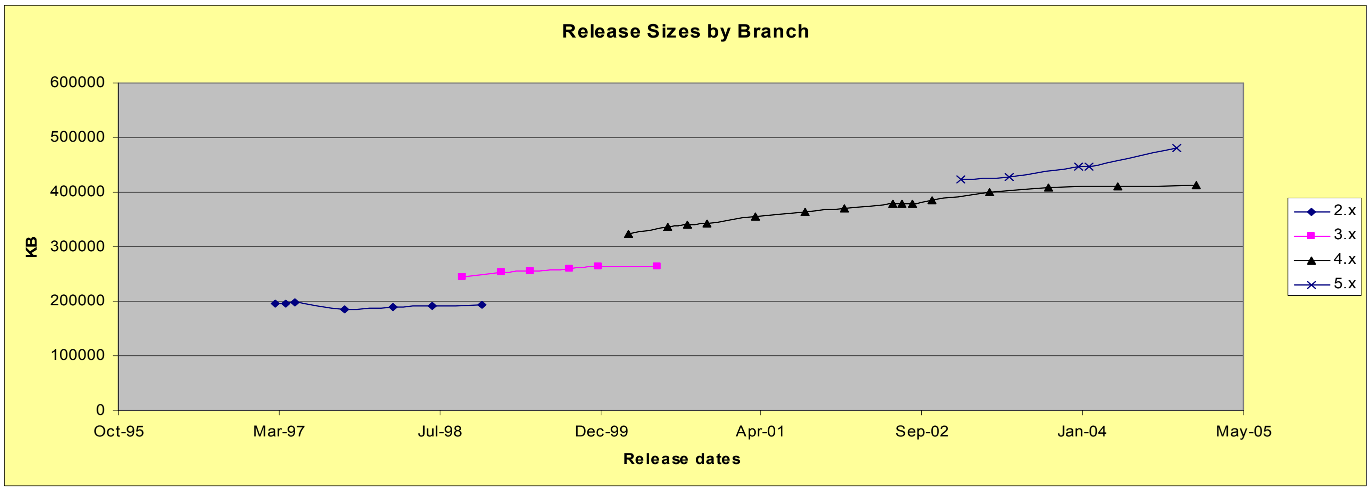

Figure 4A. FreeBSD Release sizes by development branch.

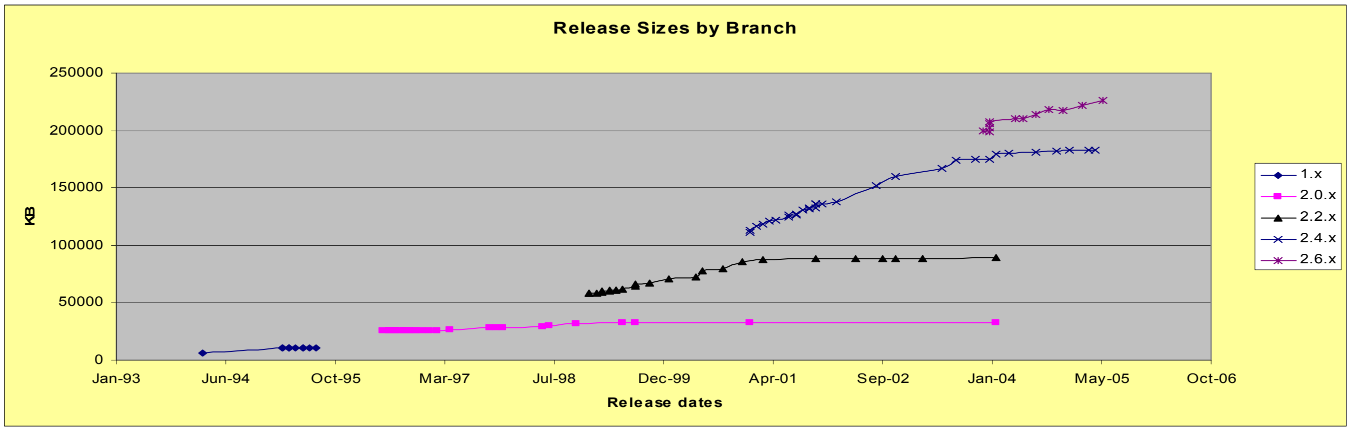

Figure 4B. Linux Release sizes by development branch.

FreeBSD branch 5.x and Linux branch 2.6.x are the only branches displaying what appears to be linear growth. Figure 4 displays graphs for FreeBSD and Linux.

We also studied the average and median sizes of dot-c and dot-h files, as shown in Figure 5. We show that, as FreeBSD and Linux continue to evolve, average and median file sizes remain constant for both dot-c and dot-h files. This may indicate that both operating systems are not growing in an uncontrolled manner.

We generated plots of the total number of C, H, C++, Makefile, scripts, directories, and LOC measure. The results are all similar. For the sake of brevity these graphs are not included, but they all show clear linear growth and seem to agree with the empirical findings of Lehman. Contrary to Godfrey and Tu, we do not find significant evidence to suggest super linear growth at the system level.

4.2 Observations at the Sub-System Level

Godfrey and Tu plotted growth of the individual sub-systems. We have done the same in this study, and have separated the sub-systems into categories of small, medium, large, and very large. We separated the sub-systems so that we could show the plots more clearly. Had we not done this, then much information could be visually lost.

Most of the sub-systems show linear or constant growth. There are some cases that show very steep spikes in growth or shrinkage; however the spikes can be attributed to the addition or removal of functionality. There were also various plots that displayed a flat line followed by a sudden spike, indicating that the system was suddenly introduced.

For FreeBSD we show two interesting and very large sub-systems separately in figure 6a. The *contrib* sub-system is software contributed by users, whereas *sys* is the actual kernel of the system and probably goes through a much stricter validation process than *contrib*. Regardless, we see linear growth in both sub-systems. For Linux, we show the growth trends of the *driver* sub-system, which mainly delineates growth caused by the expanding popularity of the system.

Godfrey and Tu found their most significant evidence of super linear growth in the *driver* sub-system. We attribute this to the increasing popularity of the operating system rather than the inherent growth properties of the software.

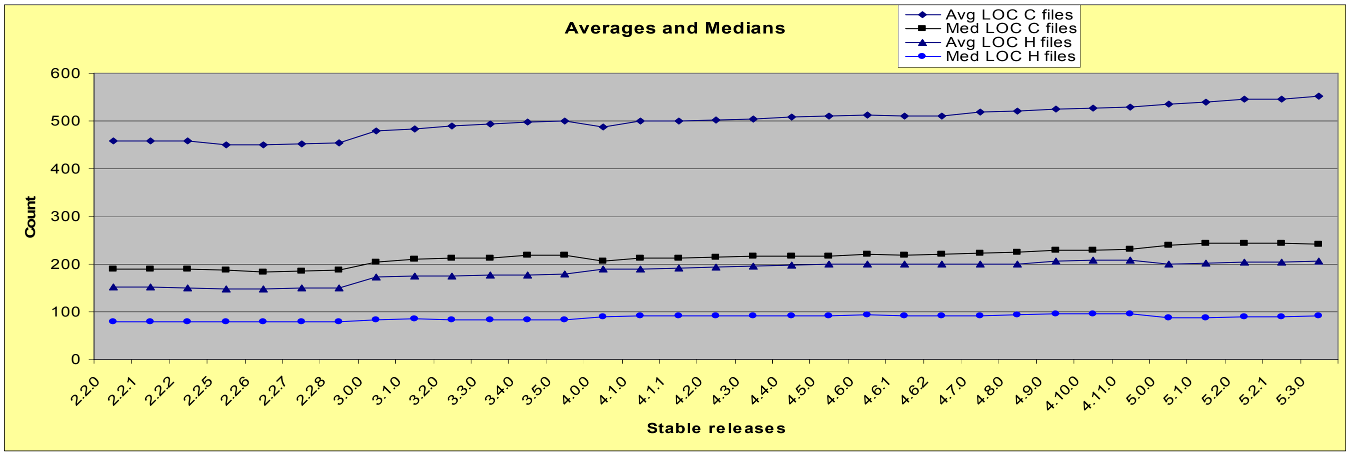

Figure 5A. FreeBSD average and median values of dot-c and dot-h files.

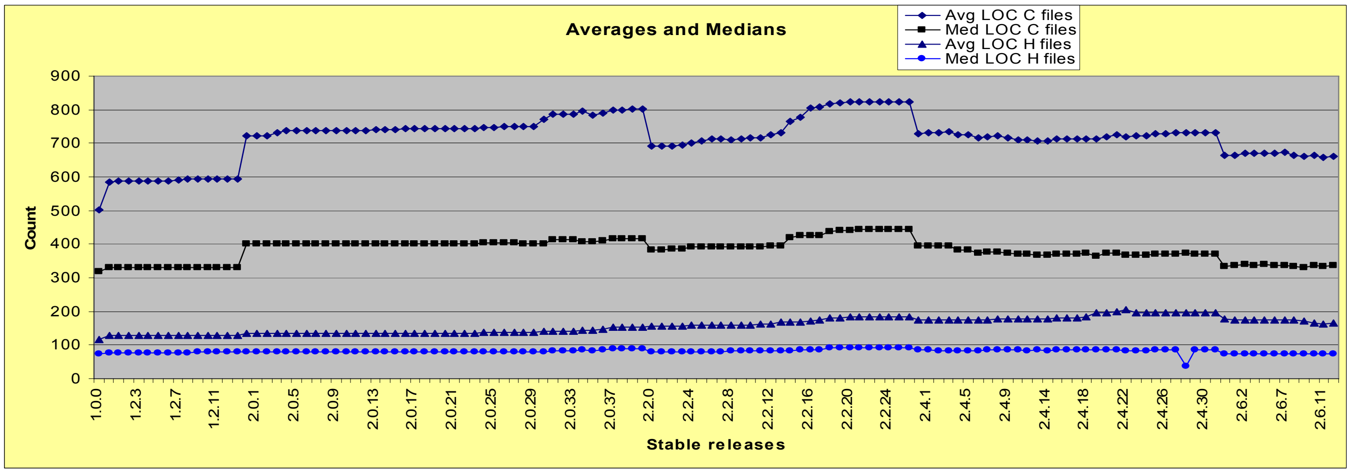

Figure 5B. Linux average and median values of dot-c and dot-h files.

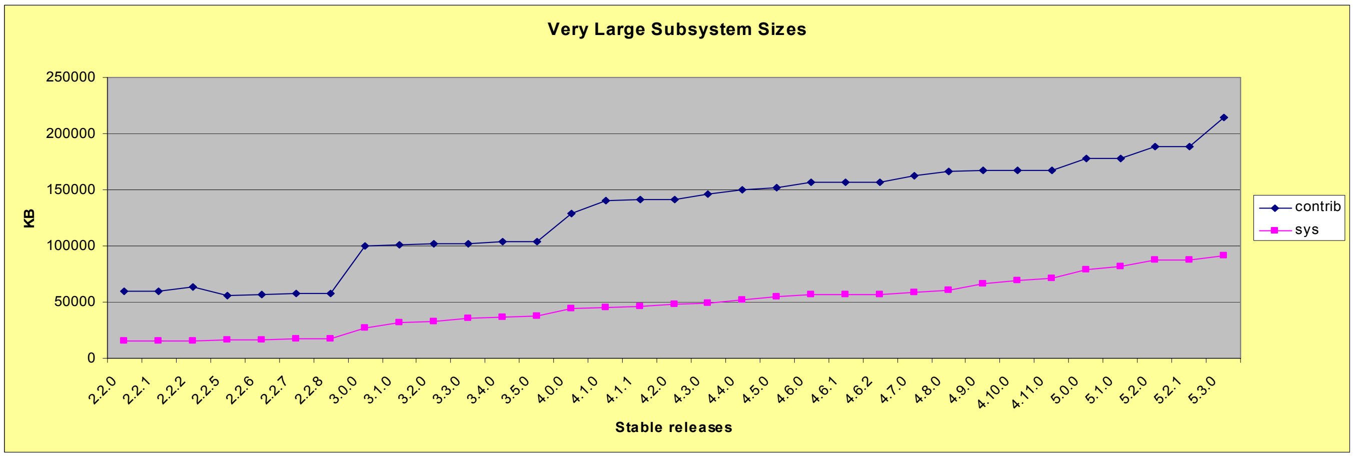

Figure 6A. FreeBSD contrib and sys sub-systems.

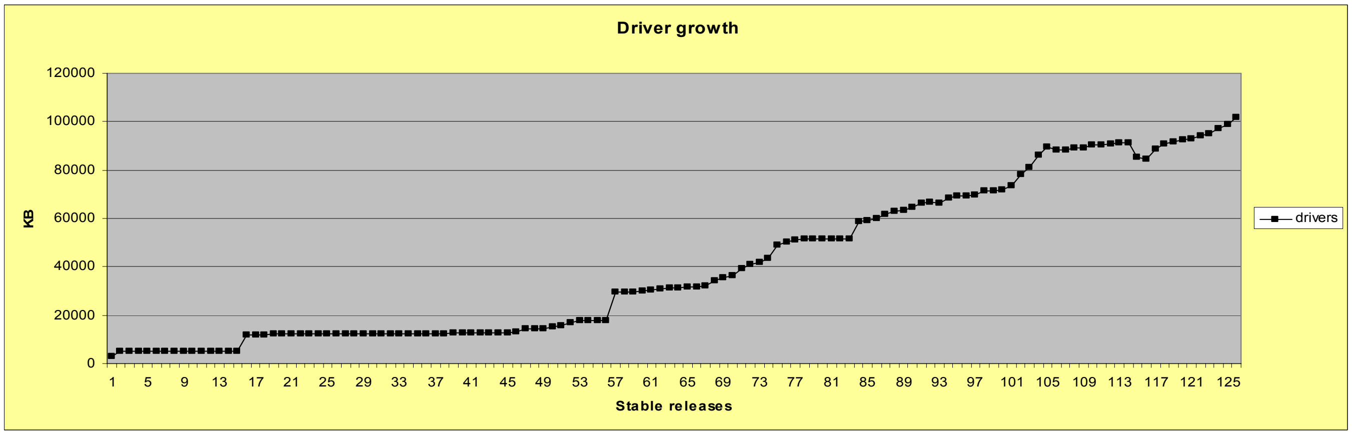

Figure 6B. Linux kernel sub-system.

5. DO OSS SYSTEMS EVOLVE AT GREATER RATES THAN TRADITIONAL SOFTWARE?

Clearly, the studies of Linux and FreeBSD do not provide evidence to refute our working null hypothesis that the growth rates of OSS and traditional systems are similar. Certainly the growth rate of FreeBSD is (approximately) linear. We find linear growth when we examine the system as a whole, or individual subsystems.

Our reexamination of Linux, using stable releases only, also shows (approximately) linear growth. One might argue that the Linux may be trending towards a super-linear curve. However, if that is the trend, the trend appears to be only slightly super-linear. The data is not strong enough to refute the null hypothesis.

An implication of this analysis is that we cannot say that OSS development produces software at a faster rate than traditional development. The evidence does not support claims that OSS systems grow faster. OSS development may offer advantages over traditional development, but we did not find support for claims concerning system growth.

6. THREATS TO VALIDITY

There are threats to validity in all case studies. We assess construct validity, content validity, internal validity, and external validity.

A study with construct validity uses meaningful measures. To have construct validity we need to know that our measures of system size and releases, actually quantify the notion of size and releases. To have content validity, the measures must completely represent the notions of size and release. There are various notions of size. We use several size measures: number of source code lines, source files, and Kbytes of source. These measures should capture the notion of system size. They expand on the notion of size used in the studies of traditional software, which used only the number of modules to indicate size. Thus we increase the content validity over prior studies. There are other notions of size, such as functionality. However, there are not adequate methods for collecting functionality data from source code.

Our use of stable releases, rather than "current" releases used by Godfrey and Tu, improves the content validity, since stable releases correspond to the releases used in traditional software. The stable releases are the ones actually "delivered" to system users.

Internal validity refers to the causal connection between the independent variables and dependent variables. In this study, there is really one independent variable, the type of development: traditional versus OSS. The dependent variable is the growth rate. There is a rationale for a dependency on development method. Traditional and OSS development use different processes and different developer motivations. However, we did not find fundamental differences in the growth rates of the two systems. Because we studied only two OSS systems, we cannot apply statistical analyses to our results.

External validity indicates that the study results can generalize to other systems. Our results on two systems from the same domain (operating systems) may generalize to other OSS operating systems. However, we cannot determine if other OSS systems will show similar results. This is a common problem with case studies. Each case study provides a new piece of evidence, and thus reduces the threats to external validity.

This study does have some specific validity threats. The data on traditional systems depends on published studies from long ago. We did not reexamine the raw data on these studies. Also, we did not examine data from more recent traditional systems. Traditional systems built over the last decade (the same time period as FreeBSD and Linux) may grow at different rates than those from before. Unfortunately, data from traditional development is difficult to obtain due to proprietary concerns.

Additional research can reduce threats to validity, especially threats to external validity. We would like to see additional studies of other OSS operating systems as well as OSS systems in other domains.

7. CONCLUSIONS

This study examined the evolution of FreeBSD and Linux to see if they show evidence of linear growth. We studied growth rate measures and plotted them against release numbers, release calendar dates, and by code branches. We tracked evolution at the system and sub-system level. In all cases we found no evidence of super-linear growth as suggested by Godfrey and Tu, and found instead that the growth rates appear to be linear or sub-linear.

Our results suggest that both FreeBSD and Linux do not exhibit growth rates that differ notably from the commercial systems studied by Lehman and others. Thus, we cannot say the OSS systems grow at a different rate than traditional systems.

This study is another data point in creating a body of evidence to continue our understanding of software evolution. Further behavioral and structural case studies are needed to further increase the body of evidence to better understand evolution.

8. REFERENCES

[1] Dinh-Trong, T., and Bieman, J. The FreeBSD Project: *A Replication Case Study of Open Source Development*. IEEE Trans. Software Engineering, 31(6):481-494, June 2005.

[2] The FreeBSD website. http://www.freebsd.org

[3] Gall, H., Jazayeri, M., Kloesch, R., and Trausmuth, G. *Software evolution observations based on product release history*. Proc. of the 1997 Intl. Conference on Software Maintenance (ICSM 1997), Bari, Italy, Oct 1997

[4] Godfrey, M., and Tu., Q. *Evolution in Open Source Software: A Case Study*. Proc. of the 2000 Intl. Conference on Software Maintenance (ICSM-00), San Jose, California, October 2000.

[5] Godfrey, M., and Tu., Q. http://plg.uwaterloo.ca/~migod/papers/icsm00-slides.pdf. Presentation at the 2000 Intl. Conference on Software Maintenance (ICSM-00), San Jose, California, October 2000.

[6] Godfrey, M., and Tu., Q. *Growth, Evolution, and Structural Change in Open Source Software.* Proc. 2001 Intl. Workshop on Principles of Software Evolution (IWPSE-01), Vienna, September 2001.

[7] Lawrence, M.J., *An Examination of Evolution Dynamics*. Proc of the 6th International Conference on Software Engineering. IEEE Computer Society Press. Sept. 1982.

[8] Lehman, M.M., *Laws of Software Evolution Revisited*. Proc of the 1996 European Workshop on Software Process Technology (EWSPT). Nancy, France, 1996 Lecture Notes in Computer Science 1149, pp. 108-124, 1997.

[9] Lehman, M.M., Ramil, J.F., and Wernick, P.D. *Metrics and Laws of Software Evolution.* IEEE Int. Software Metrics Symp., 1997.

[10] Lehman, M.M., Ramil, J.F., *Evolution in Software and Related Areas*. Proceedings of the 4th International Workshop on Principles of Software Evolution. Sept. 2001.

[11] Evolution And Software Technology (FEAST) web site. http://www.doc.ic.ac.uk/~mml/feast

[12] Lehman, M.M., *Uncertainty in Computer Application and its Control through the Engineering of Software*. J. of Software Maintenance: Research and Practice, v.1, n.1, Sept. 1989, pp. 3-27.

[13] The Linux website. http://www.kernel.org

[14] Mockus, A., Fielding, R., and Herbsleb, J. *Two Case Studies of Open Source Software Development: Apache and Mozilla*. ACM Transactions on Software Engineering and Methodology, Vol. 11, No 3, July 2002. Pages 309-346.

[15] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y. *Evolution Patterns of Open-Source Software Systems and Communities*, Proc. 2002 Int. Workshop Principles of Software Evolution, 76-85, 2002.

[16] Scacchi, W. *Understanding Open Source Software Evolution: Applying, Breaking, and Rethinking the Laws of Software Evolution*. Tech. Report. Institute for Software Research. University of California, Irvine, April 2003.

[17] Turski, W.M., *Reference Model for Smooth Growth of Software System*. IEEE Transactions on Software Engineering, 22(8), Aug 1996.

[18] The Unix History website. http://www.levenez.com/unix , maintained by Eric Levenez.