Evaluating the Evolution of Small Scale Open Source Software Systems

Chanchal Kumar Roy and James R. Cordy

Queen's University, Kingston, Ontario, Canada K7L3N6
{croy, cordy}@cs.queensu.ca
http://www.cs.queensu.ca/~cordy

Abstract. For real-world software to remain satisfactory to its stakeholders requires its continual enhancement and adaptation. Acceptance of this phenomenon, termed software evolution, as intrinsic to real world software has led to an increasing interest in disciplined and systematic planning, management and improvement of the evolution process. Almost all of the previous work on software evolution has been concerned with the evolution of large scale real-world software systems developed within a single company using traditional management techniques, or with the large scale open source software systems (LSOSSS). However, there is to our knowledge little or no work that has considered small scale open source software systems (SSOSSS). This paper presents an analysis of the evolution behavior of two small size open source software systems, the *Barcode Library* and *Zlib*. Surprisingly, unlike large scale open source software systems, the evolution behavior of these small size open source software systems appears to follow Lehman's laws for software evolution.

Key Words: Small Scale Open Source Software Systems, Case Study, Software Evolution, Metrics, Lehman's Law of Software Evolution, Software Maintenance.

1 Introduction

The software maintenance phase of the software life cycle is often underestimated, although it consumes remarkable resources. During the 1970s, maintenance was estimated to account for thirty-five to forty percent of the software budget for an information systems organization. By the 1980s, estimates of the expenditure had reached sixty percent of a software system, from initial vision to abandonment [8].

Software systems evolve during the software life cycle due to product improvement and additional development to include new features. This growth is accompanied by increasing complexity of the software. At a certain point, every new release introduces new complexity and renders development more difficult. As this point is reached it becomes necessary to analyze the structure of the system to identify potential shortcomings which may be candidates for restructuring and as a consequence, the term *Software Evolution* is introduced. *Software*

evolution, i.e. the process by which programs change shape, adapt to the marketplace and inherit characteristics from preexisting programs, has become a subject of serious academic study in recent years. Partial thanks for this goes to Lehman and other pioneering researchers. Major thanks, however, goes to the increasing strategic value of software itself [16].

The larger a software product grows, the more important the ability of the architecture of the system to support evolvability becomes. Focusing on the analysis of software evolution has revealed a high potential for improvement of the software development process. Software evolution is more important as systems become longer lived. However, evolution is challenging to study due to the longitudinal nature of the phenomenon in addition to the usual difficulties in collecting empirical data [8].

Often the real structure of software system is not fully known to engineers. Therefore, it is difficult to reason about any possible deficiencies and how they may be corrected. Thus, it is critical to capture software architecture for the maintainability of a software system. After such an analysis, restructuring can be applied, which may help to improve the discovered situation. In the last few years numerous methods for the analysis of software architecture have been developed. Different sources of information, for example release information, design documents, and source code, are used to draw conclusions about the structure of software systems. Many approaches base their analyses on the micro level of program source code. For very large systems these approaches can reach a level where it is difficult to make any reliable statements about the quality of the software. Instead, methods that focus on the macro level (for example number of modules/subsystems, number of classes, and so one) must be applied to analyze these large systems.

A study of the literature reveals that most of the previous work on software evolution has looked primarily at large real-time software systems [11] or large open source software [13], [14], such as the Linux kernel. Surprisingly, little or no work has been concerned with the evolution of smaller size software systems, which leads us to think about observing the evolution behavior of small size open source software systems. This paper presents a study of the evolution behavior of two such small size open source software systems, the *Barcode Library* and *Zlib*. We consider the evolution for both the systems, with particular emphasis on the *Barcode Library*. We have applied both micro and macro level analysis to these systems. However, as our target systems are very small in size, we particularly focus on the micro level, measuring evolution of the uncommented non-blank line of codes (UNBLOC or ULOC for short).

The rest of this paper is organized as follows. In Section 2, some significant related work is reviewed. In Section 3, a brief description of the target open source software systems is provided. Section 4 presents the methodology and measuring metrics used in this work. In Section 5, a detailed case study observing the evolution behavior of the target software systems is presented, and finally Section 6 summarizes our observations and conclusions.

2 Related Work

An extensive study on software evolution has been carried out by Lehman et al. [9], [10], [11], [12], over the last 35 years. Based on their experience we have several laws of software evolution, known as Lehman's Laws of Software Evolution [11], [12]. The work by Godfrey et al. [7] has shown that Linux continuously exhibits a global super-linear growth pattern, and recently Robles et al. [14] conclude that Lehman's laws, especially the 4^{th} law [11] is not well fitted to large size open source software systems. However, a more detailed empirical study [13] does not support the hypothesis that open-source development fosters faster system growth than closed-source development as opposed to [7], [14]. In [6], the authors examined the structure of several releases of a Telecommunications Switching System (TSS) stored in a database of product releases and found that there is a significant difference in the behavior of the whole system versus its subsystems. In [2], Burd et al. have evaluated the evolution of the GCC compiler and found interesting results. Capiluppi et al. have also authored several works on the evolution of open source software projects [4] and have proposed some models [3] working with mid-size projects. Succi et al. [17] also observed super-linearity in the evolution of the Linux kernel, but found linear growth for both GCC and Apache.

3 Target Open Source Software: *Barcode Library* and *Zlib*

In our work we have studied two useful small scale open source software systems, the *Barcode Library* and *Zlib* Tool. The Barcode package [1] is mainly a C library for creating bar-code output files. It includes both command line and graphical front-ends. The package is designed as a library because the main use for barcode-generation tools is embedding in other applications. The library addresses bar code printing as two distinct problems: creation of bar information and actual conversion to an output format. Based on the functionality, the *Barcode Library* is divided into five modules: the *Front-End Module*, the *Output Module*, the *Basic Encoding Module*, the *Advanced Encoding Module* and the *Header File Module*. More detail about Barcode can be found in [1] and in the relevant documentation of the different versions.

Zlib [15] is a well known compression tool which is also designed to be a free, general-purpose, legally unencumbered (i.e., not covered by any patents), lossless data-compression library for use with virtually any computer hardware and operating system. The *Zlib* data format is itself portable across platforms. In this work we study both of these tools but focus particularly on the evolution of the *Barcode package*.

4 Methodology

We have collected 9 different releases of the *Barcode Library* and 43 different releases of *Zlib* from their home pages. We have calculated the total size, total

lines of code (LOC), and number of uncommented non-blank lines of code (UNB-LOC or ULOC for short) for each of the releases of both the software systems. For the case of *Barcode Library*, we have also done the same for the five modules mentioned in Section 3 above. The open source program Numlines [18] is used to assist in these calculations. For each of the releases of the *Barcode Library* and *Zlib* we have also calculated the number of global functions, variables and macros using Ctags [5]. For drawing the least-squares fit of the plotted data a small java tool is developed and called the Least-Squares Fitter (LSF).

There are several kinds of metrics being used by different researchers. In this work we have used the following metrics for observing the evolution of *Barcode Library* and *Zlib*. In the following, we use the usual notation Release$_i$ for the i^{th} release, Growing Rate$_i$ for the *growing rate* of the i^{th} release, and so on.

Size$_i$ = Size of Release$_i$ in bytes i.e., total size of ith release in bytes
UNB-LOC$_i$ = Number of uncommented non-blank LOC in Release$_i$
Added$_i$ = Number of UNB-LOC of the newly added files in Release$_i$
Changed$_i$ = Number of UNB-LOC of the changed/modified files in Release$_i$
Growing Rate$_i$ =(Added$_i$ *X* 100) / (UNB-LOC$_{i-1}$)
Changing Rate$_i$ =(Changed$_i$ *X* 100) / (UNB-LOC$_{i-1}$)
Handled$_i$ = (Changed$_i$ + Added$_i$) - (Changed$_{i-1}$ + Added$_{i-1}$)
UnHandled$_i$ = (UNB-LOC$_i$)- (Handled$_i$)
Handling Rate$_i$ =(Handled$_i$ *X* 100)/ (UNB-LOC$_i$)

Once the data was collected for each of the releases of *Barcode* and *Zlib* libraries, we plotted the data over time, for *Zlib* against the release serial number (RSN) as suggested by Lehman, and for the *Barcode Library* against calendar time (number of months/days since first release) following the style of several other researchers. We then considered whether our plotted data appears to follow Lehman's laws of software evolution [11] or not.

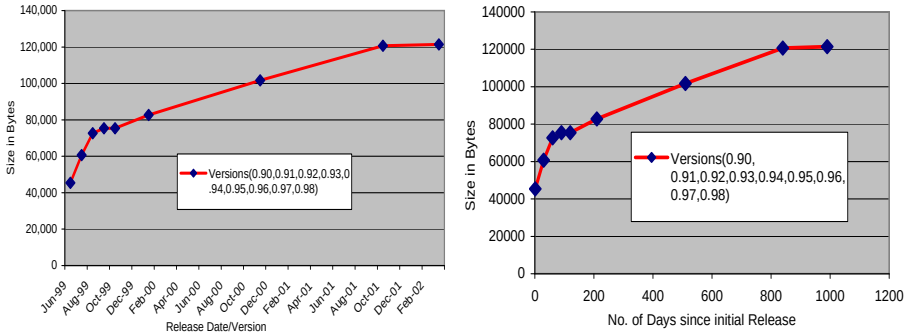
5 Observations on the Evolution of *Barcode Library* and *Zlib*

During the analysis of the software systems we have made some interesting observations. In the following subsection we concentrate on the global systems and their evolution first. Then we zoom in on the subsystems. The outliers that may be recognized in the subparts are discussed in more detail. The evaluation covered releases of the *Barcode Library* from June 1999 until March 2002, which contains 9 releases over the period of thirty three months. For *Zlib*, we considered 43 releases covering a period of about 11 years.

5.1 Overall Growth w.r.t. Size

To begin, Fig. 1(a), shows the overall growth of *Barcode Library* beginning from the first version released in June 1999. We can see that over time the size of the system has increased to meet additional user requirements. We have manually

analyzed the *Barcode Library* documentation for all of the versions and found that in most of cases the growth was required to add new functionally to meet user requirements. Thus later versions have more functional capability, covering more user needs. In the early days of the *Barcode Library* there were a number of releases with fewer new functionalities added, and therefore we can observe (Fig. 1(a)) a period of several versions with comparatively little growth in size compared to the first three. However, we see that the growth follows a monotonic process and therefore we can say that the growth of *Barcode Library* does follow Lehman's 6^{th} Law of continuing growth of system size.



(a) Over Months (b) Over Days

Fig. 1. Growth of Barcode Over Time (months + days) w.r.t Size of the Released Versions

This is even clearer in Fig. 1(b), where we have plotted the size data against the number of days since the first release. We have used our LSF tool to compute a least squares fit of the data and found that the linear equation $Y = 65.92X + 63138.28$ where X is the number of days, is a good predictor of Barcode system size over time. This linear growth is in contrast to the super-linear growth of Linux, reported in [7], [14], and can be termed continuing growth, Lehman's 6^{th} law of evolution.

In Fig. 2, we have created a similar plot for *Zlib*, but in this case we have plotted the data against the release serial number (RSN) as suggested by Lehman rather than time. Here, we also find that system growth continues over time. We have analyzed the changed log histories of the different versions and also found that, like Barcode, the growth seems to be primarily due to the addition of more functionalities to meet additional user requirements.

As we see, even when plotted against RSN, growth of this system also seems to be liner in nature and can be fitted to the linear equation $Y = 6568.53X + 175518.76$ with correlation coefficient 0.9668. Therefore, we can say that the growth of *Zlib* also follows the Lehman's 6^{th} law of continuing growth. However,

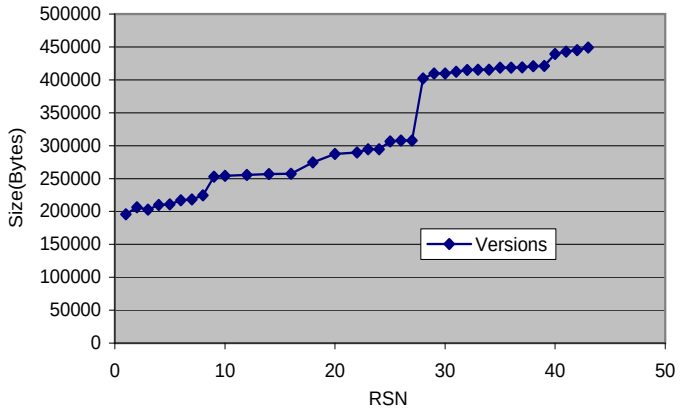


Fig. 2. Growth of *Zlib* Over RSN w.r.t Size of the Released Version

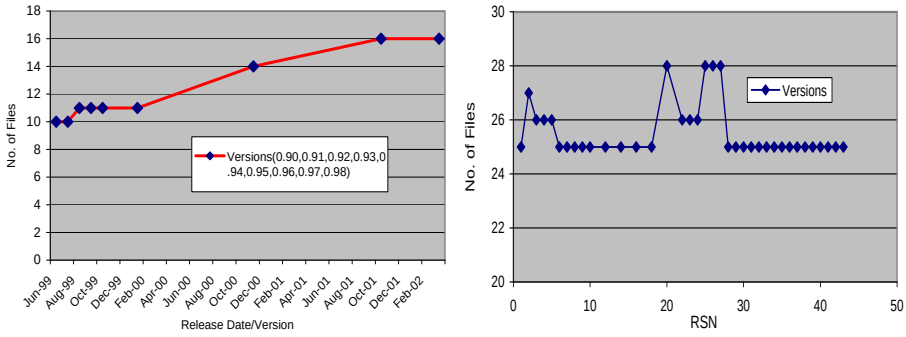
we can also see that although growth is continuing, occasionally it is very high compared to other releases, as we also observed in the case of the *Barcode Library*. In both cases we found similar growth with respect to size of the development releases, and both seem to follow Lehman's 6^{th} law of evolution.

5.2 Growth w.r.t. Number of Files

Measuring growth with respect to the size of released versions might be questionable in the sense that a system could grow in size without adding new functionalities, for example due to addition of unnecessary lines of code (e.g., cloned or redudant code added by a poor coder) or a large number of new comments or blank lines. For this reason recent researchers often use other kinds of metrics, such as number of modules, number of subsystems, number of classes and so on, to measure object-oriented source code. As our target systems are written in C and are quite small in size, we have considered number of files as another of our metrics.

In Fig. 3(a), the number of files in the releases of the *Barcode Library* have been plotted against calendar time. For this system it seems that there has been little significant growth in the number of files, and only 6 files have been added over the entire period of evolution from the initial version to the latest version

In Fig. 3(b), we have found more interesting results for *Zlib*. In this case we see that the number of files remains relatively constant over all releases, except for some earlier releases where the size of the system has actually been decreased over time. In the case of Barcode (Fig. 3(a)), we could say that system is growing continuously in number of files over time, following the 6^{th} law of Lehman, but the same is definitely not true in the case of *Zlib* (Fig. 3(b)) where it is clear that number of files remains constant over most of the latest releases. This is very much contrasted with the literature, especially for large open source



(a) Barcode (b) Zlib

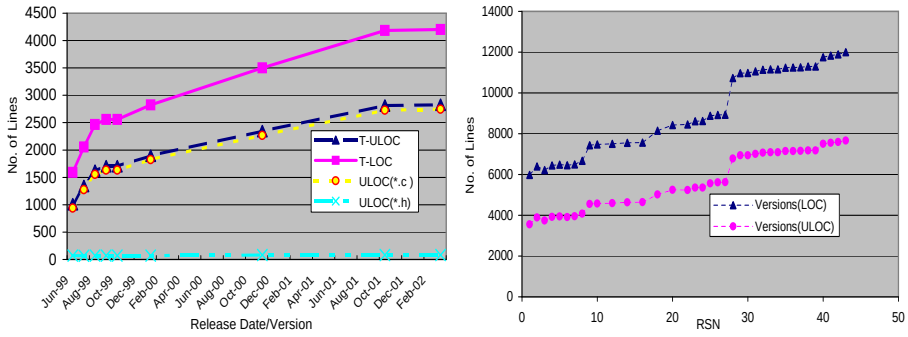
Fig. 3. Growth Over Time (Months) w.r.t the Number of Files

systems, where the reported growth has even been super-linear with respect to the number of modules/subsystems [7], [14]. We therefore decided to consider other kinds of metrics as well.

5.3 Growth w.r.t. LOC and UNB-LOC

Considering the unusual (lack of) growth in *Zlib* with respect to the number of files as shown in Fig. 3(b), we decided that perhaps simply counting the number of files might not be a good metric for evolution of these systems. When target systems are very small, perhaps in most cases the number of files remains more or less the same over the evolution period. We have considered the number of lines over system releases. As the total number of lines may also contain commented lines or blank lines, we decided to try observing evolution by considering only the number of uncommented, non-blank lines of code (UNB-LOC). UNB-LOC is an old technique for measuring evolution, but we have found that for small size software systems, especially those are written in non-object-oriented context, considering the UNB-LOC may be a better or perhaps even the only appropriate metric for evolutionary growth.

In Fig. 4(a), we have plotted the total lines of code (T-LOC) and total UNB-LOC (T-ULOC) for the *Barcode Library*. We have also shown the total UNB-LOC for the *.c files and *.h files separately. As we see, there is a linear growth over time except in the case of the *.h (header) files. Our main concern is the T-ULOC of the releases and we see that the growth of this curve is a good fit to the linear equation $Y = 44.55X + 1115$. This leaves little doubt that this measure of growth of releases of this system seems to follow Lehman's 6^{th} law of evolution The LOC and UNB-LOC (ULOC for short) for *Zlib* have also been plotted in Fig. 4(b) against RSN and we can observe the same linear nature of continuing growth. In the case of the UNB-LOC curve, the linear equation



(a) Barcode (b) Zlib

Fig. 4. Growth Over Time (months) w.r.t. T-LOC and UNB-LOC

$Y = 103.41X + 3356.25$ is found to fit with correlation coefficient 0.977, definitely consistent with Lehman's 6^{th} law of evolution.

5.4 Handled and Unhandled UNB-LOC for the *Barcode Library*

Until now we have discussed only the 6^{th} law of evolution. However, what fraction of the system is handled or unhandled is also a major factor in software evolution and is directly related to some of the Lehman's laws. In this section we have studied what fraction of the system is handled or remains unhandled over time, in order to estimate the *changing rate* and *growing rate* of the system.

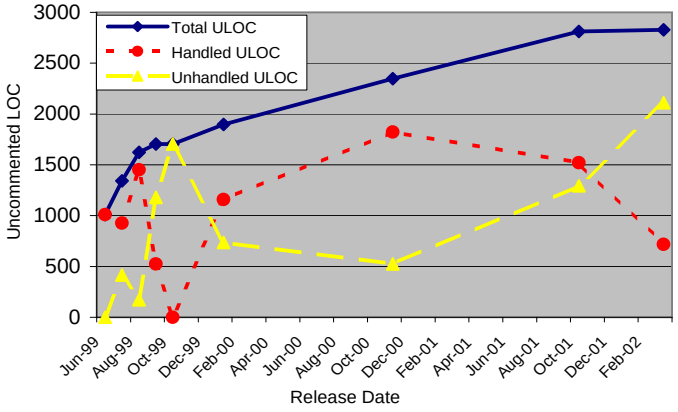


Fig. 5. Handled and Unhandled UNB-LOC Over Time (months) for Barcode.

In Fig. 5, we have plotted the ULOC, the handled ULOC, and *unhandled ULOC* for the total system. We have found interesting results for the case of *handled ULOC*. We see that except for some early releases, the *handled ULOC* consistently increased for a time and since then has decreased significantly. We can also see the same effect in the corresponding *unhandled ULOC* curve. The *handled ULOC* and *unhandled ULOC* curves can be fitted with the linear equations $Y = 11.58X + 892.20$ and $Y = 36.48X + 519.19$ respectively.

One interesting observation to note here is that there is no change from version 0.93 to version 0.94 (during the time period October 1999 to December 1999), as we see that the *unhandled ULOC* curve touches the total ULOC curve and *handled ULOC* curve is at the X-axis. It would be interesting to find out how and why version 0.94 evolved from version 0.93.

Next, in Fig. 6 we have plotted the number of files that are changed and newly added over time (although we know number of files is not a good metric for such small size software systems).

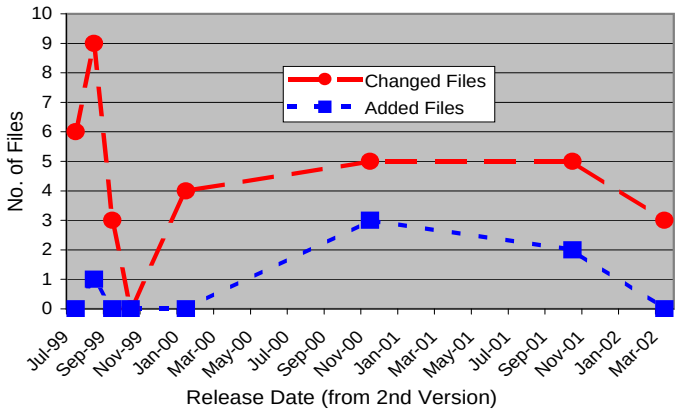


Fig. 6. Changed/Added Files Over Time(months) of Barcode.

We see that very few files are added over time and this adding phenomenon is also decreasing in the latter versions. However, initially the number of changed files is more than in later versions, and the number is also decreasing, following some properties of Lehman's 1^{st} and 5^{th} laws. We know Lehman's 1^{st} law predicts continuing change and the 5^{th} law concerns conservation of familiarity. To focus on these two laws, we have plotted the *changing rate* and *growing rate* of *Barcode* in Fig. 7 by taking into account the *handled ULOC*. In this Fig. 7, we see that the *Barcode Library* system is changing continuously over time, with some exceptions in the earlier versions, especially in the case of version 0.94 from version 0.93. Therefore, we can say that this changing nature of *Barcode Library* follows the Lehman's 1^{st} law. On the other hand, both the *changing*

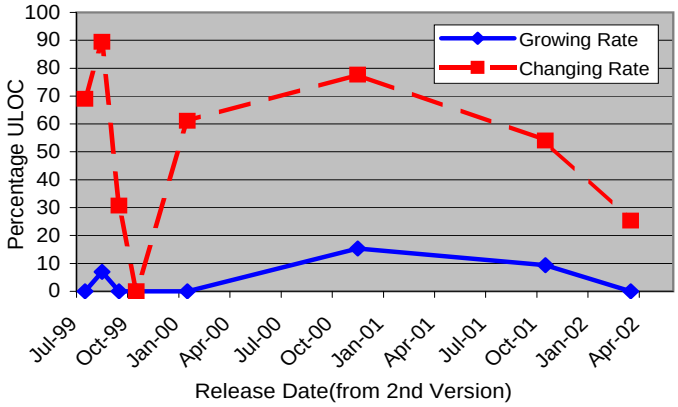


Fig. 7. Growing Rate and Changing Rate of Barcode Over Time (months).

rate and *growing rate* tend to decline over time which lead us to the 5^{th} law of evolution, the *Conservation of Familiarity*. However, as we mentioned earlier, we really could not predict what actually added/modified to version 0.94 from its previous version 0.93 and therefore, in the next subsection, we go into more depth on the *Barcode Library* by observing its evolution at the module level.

5.5 Module Level Observations of the *Barcode Library*

As we know from Section 3, five modules are considered in the *Barcode Library* corresponding to parts of the two main purposes of the library. To observe the evolution of the module level, we have plotted the UNB-LOC for each of the five modules of the system over time in Fig. 8. We see that except for the *Header File module*, all others are increasing in the number of UNB-LOC over time, following Lehman's 6^{th} law of continuing growth and signaling the 1^{st} law of continuing change. We have found that the most promising module is the *Basic Encoding Module* which is increasing more rapidly over time than any other module with respect to both size and rate of growth. We have also noticed that except for the *Basic Encoding Module*, all other modules tend to decline in incremental growth which brings us in mind of Lehman's 5^{th} law of evolution on the module level also. We can therefore say that with few exceptions, the evolution of the Barcode Library modules follows Lehman's 1^{st}, 5^{th} and 6^{th} laws of evolution well.

To be sure about our predictions, in Fig. 9 we have plotted the percentage UNB-LOC of the modules with respect to the total system over time. Here we see that except for the *Advanced Encoding Module*, all other modules either change in parallel or tend to decline in incremental size with respect to the percentage of the total system. A parallel curve, like the later releases of the *Basic Encoding Module*, implies that this module is growing at the same rate as the total system. However, we can also see that except for the *Advanced Encoding*

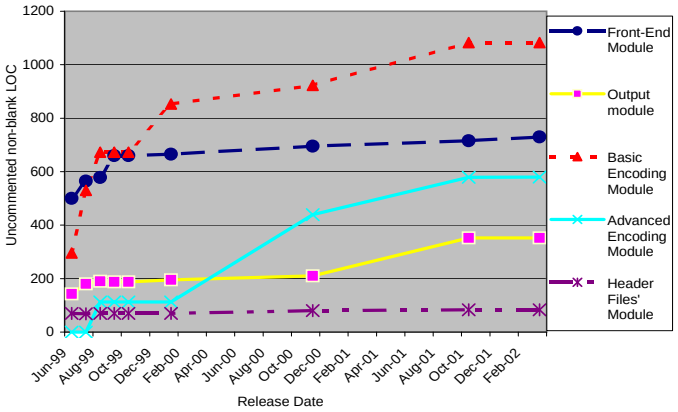


Fig. 8. UNB-LOC of the Modules of Barcode Over Time.

Module, all others are decreasing in their rate with respect to the whole system, demonstrating the predictions of Lehman's 5^{th} law.

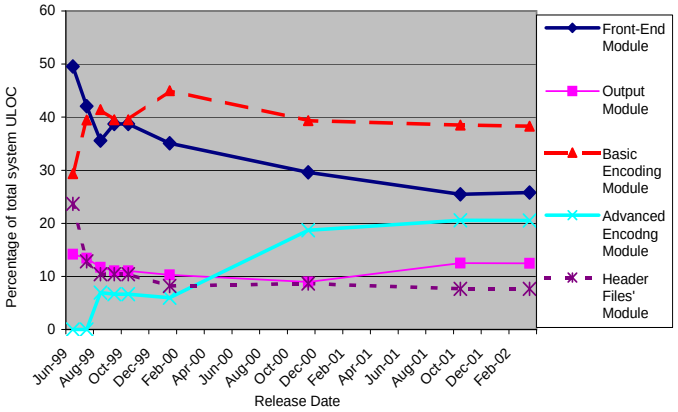


Fig. 9. Percentage of ULOC of the Modules w.r.t the Total System Over Time.

We mentioned earlier the problem with version 0.94, where we could not figure out how or what had actually been changed or added in version 0.94 from version 0.93. We have assumed that something might happened in the *Basic Encoding Module* as this seems the most promising module among the others. Therefore, we have done a more in-depth focus on this module by plotting its individual files in Fig. 10. As we can see from this figure, files *code39.c* and *code128.c* are growing over time, while *ean.c* is not. We can also see that

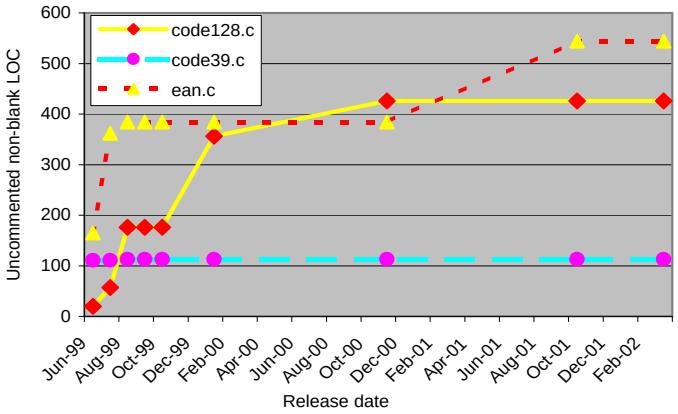


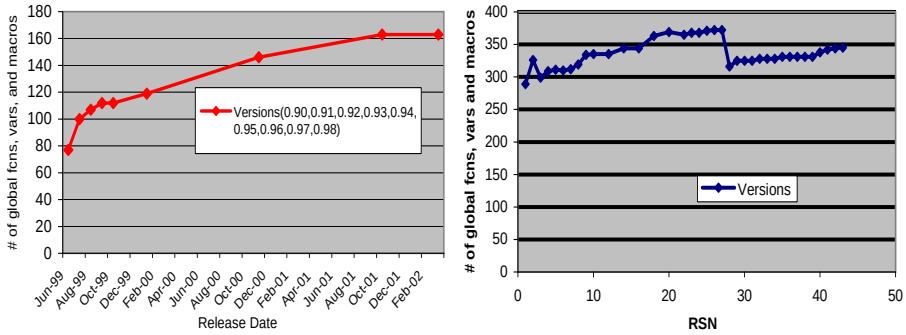
Fig. 10. File-Level Growth of the *Basic Encoding Module* Over Time of Barcode

code128.c has a smoother growth than the others, which helps us imagine that *code128.c* might be one of the main files of this module, or even of the whole system (since *Basic Encoding Module* is the most growing module and *code128.c* is in this module). Perhaps it changes over versions at a smooth rate and most of the system changes are being made in this file.

However, we still cannot predict clearly how and why version 0.94 evolved from version 0.93 as yet, even with these file level observations. With our strong belief that evolution might be associated with *code128.c* (as most promising file, in most growing module), we have considered the size of this file over versions 0.93 to 0.94 and found that there are only 2 bytes of difference between these two files. Even more interesting, there are only 2 bytes of overall size difference between versions 0.93 and 0.94. For our further interest, we have used the Linux command *diff* to see the real differences between *code128.c* of version 0.93 and that of version 0.94, and we have found that a few lines of code have been replaced with the same number of lines but with more complicated code in the same lines of *code128.c* in version 0.94 from version 0.93. We have also found that some comments are added with the same lines.

5.6 Observing the Complexity

Complexity of a system is always a major concern. Lehman also has the 2^{nd} law of evolution, relating to the complexity of the system. The 2^{nd} law states that as a system evolves the complexity of the system increases unless work is done to maintain it. As we know, since the C language has a single flat namespace, the complexity of a C program depends largely on the number of global functions, variables and macros (unlike C++ or Java). We have counted the total number of global functions, variables and macros for each of the releases of Barcode and *Zlib* using Ctags [5]. We have plotted this for *Barcode Library* over time in Fig. 11(a) and that of *Zlib* against RSN in Fig. 11(b)).



(a) Barcode (b) Zlib

Fig. 11. Number of Global Functions, Variables and Macros Over Time of Releases

From Fig. 11(a), it can be seen that as Barcode has evolved, the complexity of the system has increased, following the Lehman's 2^{nd} law of evolution. We have also found a good fit to the linear equation $Y = 2.22X + 98.62$ using LSF, which indicates that the complexity of the system has been increased with roughly linear growth over the months since its first release in June, 1999.

We can also observe more or less same behavior in the case of *Zlib* in Fig. 11(b). Here it can be seen that initially the complexity of the system increases, consistent with the 2^{nd} law of evolution, and then later special work has been done to control the complexity. As we result, after the 21^{st} release, the complexity of the system decreased drastically, and then again increased gradually over the next releases, again following Lehman's 2^{nd} law of evolution.

6 Conclusion

In this work, the evolution behavior of two small scale open software systems has been observed. It has been found that the evolution of these small scale software systems is consistent with Lehman's laws of software evolution, unlike the reported evolution of larger open source software systems, which have been reported to show evolution behaviors inconsistent with Lehman's laws. In our study we have mainly focused on Lehman's 1^{st}, 2^{nd}, 5^{th} and 6^{th} laws of evolution and found that for the most part the evolution of both the *Barcode Library* and *Zlib* follows these laws with some minor exceptions.

While more research would be required to make any firm conclusions, these observations lead us to believe that perhaps small size open source software in general may follow Lehman's laws of evolution more consistently than do larger systems such as those reported by Robles et at. [14] and Godfrey et al. [7]. However, there are critical differences in our studies, including both that Robles and Godfrey have worked with very large scale open source software systems

developed by very large groups of people, and that they have concentrated on the 4^{th} law of evolution. Although we have not specifically focussed on the 4^{th} law, from the observations of this case study, it is however clear that neither *Barcode Library* nor *Zlib* has super-linear growth in their evolution and hence, it can be concluded that the evolution of both *Barcode Library* and *Zlib* does follow Lehman's 4^{th} law of evolution in some sense.

References

1. The Barcode: http://www.gnu.org/software/barcode/barcode.html
2. Burd, E., Munro, M.: Evaluating the evolution of a C application. Internation Workshop on Principles of Software Evolution(1999), pp. 401-410, Fukuoka, Japan.
3. Capiluppi, A.:Models for the evolution of os projects. Proceedings of the International Conference on Software Maintenance(2003), pp. 65-74, Amsterdam, The Netherlands.
4. Capiluppi, A., Morisio, M., Lago, P.: Evolution of understandability in oss projects. Proceedings of the 8th European Conference on Software Maintenance and Reengineering(2004),pp. 58-66, Tampere, Finland.
5. The Ctags: http://www.die.net/doc/linux/man/man1/ctags.1.html
6. Gall, H., Jazayeri, M., Klösch, R., Trausmuth, G.: Software evolution observations based on product release history. Proceedings of the International Conference on Software Maintenance(1997), pp. 160-169, Berlin, Germany.
7. Godfrey, M. W., Tu, Q.: Evolution in Open Source software: A case study. Proceedings of the International Conference on Software Maintenance (2000), pp. 131-142, San Jose, California.
8. Krajewski, J.: QCR - A Methodology for Software Evolution Analysis. Master Thesis(2003), Technical University of Vienna.
9. Lehman, M. M., Belady, L. A.: Program Evolution: Processes of Software Change, Academic Press, 1985.
10. Lehman, M. M., Perry, D. E., Ramil, J. F.: Implications of evolution metrics on software maintenance. Proceedings of the International Conference on Maintenance (1998), pp. 208-217, Bethesda, Maryland.
11. Lehman, M., Ramil, J., Wernick, P., Perry, D.: Metrics and laws of software evolution - the nineties view. Proceedings of the Fourth International Software Metrics Symposium(1997), Portland, Oregon.
12. Lehman, M. M., Ramil, J. F.: Rules and tools for software evolution planning and management. Annals of Software Engineering, **11**(2001):15-44.
13. Paulson, J. W., Succi, G., Eberlein, A.: An empirical study of open-source and closed-source software products. Transactions on Software Engineering, *30*(4): 246-256, April 2004.
14. Robles, G., Amor, J. J., Gonzalez-Barahona, J. M., Herraiz, I.: Evolution and Growth in Large Libre Software Projects. Proceedings of the International Workshop on Principles of Software Evolution(2005), pp.165-174, Lisbon, Portugal.
15. Roelofs, G., Gailly, J., Adler, M.: The Zlib home page. http://www.Zlib.net/
16. Salon: http://www.salon.com/tech/feature/
17. Succi, G., Paulson, J., Eberlein, A.: Preliminary results from an empirical study on the growth of open source and commercial software products. In EDSER-3 Workshop(2001), co-located with ICSE 2001, pp.14-15 Toronto, Canada.
18. The Numlines home page. http://www.gammadyne.com/cmdline.htm, 2005