Software Evolution Analysis of Linux (Ubuntu) OS

Mujahid Tabassum
Faculty of Engineering and Computing Science
Swinburne University of Technology
Kuching, Malaysia
mtabassum@swinburn.edu.my

Kuruvilla Mathew
Faculty of Engineering and Computing Science
Swinburne University of Technology
Kuching, Malaysia
kmathew@swinburne.edu.my

***Abstract*— Now a days, software industry is growing very fast due to advancement in Information Technology. Users, in turn, are expecting more commitment from the industry, which brings large research attention into software engineering. Software engineering is facing big challenges in software evaluation and reliability techniques and it has become one of attractive research areas. "Software engineering is a process of designing, developing and maintaining the software". Many kinds of software evaluation techniques have been proposed, but as the software complexity increases, many of them face new challenges. In this paper, we have analysis the growth and changes in 'Ubuntu Linux' Operating System (OS). Ubuntu is an Open Source OS and we have performed a quantitative analysis on various distributions of Ubuntu to analyze its complexity and growth throughout the time. Additionally, typical patterns of Ubuntu's evolution and features, their main purposes are discussed.**

Keywords-Linux; Ubuntu; Evolution Metrics; Gini Coefficient;

I. INTRODUCTION

As the technology is growing, the demand for reliable and quality software also increasing. In today's market developing reliable software is also important for many other requirements such as future growth, robustness, and standardization and user adoption. Software industry has become very competitive due to increasing demand and adaptation in every field. They really need to be diligent about software good processes for production, quality control, future growth and stability of their products. As time is passes, user expectations are rising, and also leading towards integration and synchronization of applications and services. Software companies are trying increasingly harder to satisfy user expectation by offering maximum features and functionalities in their products. Software evolution gives rise to increased complexity, causing more performance degradation and end of software life. Thus, a good software evaluation process is needed to understand the software growth and complexity on timely basis.

In today market, software is not as simple as before, the software applications are no longer stand alone systems, but the are inter-operable and interactive systems. The software applications need to have ability to work with different vendors libraries, over computer networks and many components integrated together to obtain required outputs. These software and components could be changed independently, rapidly and make difficult to manage whole system consistency, which increase software complexity and require a proper evaluation process. Therefore, software evolution and deployment process employed to understand the software growth rate and complexity over the certain period of time by using software metrics [1]. In a software engineering context, it is very important to measure the software processes of required product and that could help to understand the current state of software system [1]. "You can't control what you can't measure" was said by Tom Demarco a software engineer [2].

In this paper we evaluate the Linux (Ubuntu) distributions by analyzing its change log, metrics values and growth analysis by using Gini Coefficient parameters [3]. We investigate the evolution process of the Open Source Software (Ubuntu) from its initial release 4.0 to the latest version 13.10. The Ubuntu version of Linux is free and no licensing cost required and as Open Source License, it does not have restriction on its use, source code and relevant information for detailed investigation is available.

II. SOFTWARE EVOLUTION

Software Evolution related to the changes occurs in single software over a certain period of time. In today's age software evolves rapidly due to high competition in the industry. Lehman said that software evolution is a dynamic behavior of programming system as they are maintained and enhanced over their lifetimes, which means evolution related to maintenance activity that cause changes and focus on outcome rather than the process [4].

Originally Swanson proposed the concept of software maintenance that refers to software changes and evolution. Later on this process was updated in ISO 14764 with some mutually special activities such as i) Corrective work, to correct identified errors, ii) Adaptive work, to confirm that the software can stay stable and easily adopt future changes, iii) Perfective work, related to ensuring future growth and performance objectives and iv) Preventive work, to understand the potential system faults [5]. Chapin refines this model into 12 further orthogonal drivers, which cause software evolution such as evaluation, consultive, training, updative, reformative, adaptive, performance, preventive, groomative, enhancive, corrective and reductive. Software evolution, therefore refers to measuring the changes that occur in the software

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

from it first release to latest release in order to enhance and maintain its functionality over time [1].

III. The Laws Of Software Evolution

The term software evolution was first proposed by Dr. Meir Lehman in 1980[4]. The basic purpose of software is to provide an automation solution to user problems and reduce the human effects in order to maximize output. These problems are not static and continuously change over the time. In order to keep these solutions up to date, it needs to be modified consistently. Lehman's laws describe "the software evolution as a force responsible for driving new and revising developments in a system" [4]. Lehman categorizes all software systems into three types, S-Type (static systems for which solutions are well understandable), P-Type (practical systems for which solutions are not immediately apparent), and E-type (it refers to embedded systems which characterize the majority of daily use software) [6]. Lehman further describes 8 more laws for revolution of a software system based on their size and complexity of growth. Its obvious that when software system size grows as it is continually adapted, these systems also become more complex and require additional resources to preserve and simplify their structure. Furthermore, it suggests that when software system grows they become more challenging to modify unless special parameters are added to improve their maintainability [6].

A. Growth Dynamics

There are three kind of growth rate measurement that are used to observe evolution in software systems such as Sub Liner Growth, Super-Linear Growth and Liner Growth [1].

B. Change Dynamics

In today market, the software systems that are commonly used in the industry need to be change continuous else they will lose the user intension and interest. Changes can be identified as unit of change or integration of components. The detection of changes can be done by looking at the changes in the classes and method. This can be understood by analyzing transaction logs such as CVS logs, issue logs etc. [1].

C. Measuring Evolving Softwares

Due to tremendous growth in software industry, software's are required daily maintenance to meet the user expectation, which cause of evolving software's. Therefore, software developer / companies need to envisage long term software evolution on quantified basis.

Size and Structural complexity are two type of metrics which can be used to measure the software internal structure, size and processes. These metrics are valid and useful to measure the software evolution process but still as software size grows they become less affected to measure size and complexity of software system [1].

IV. The Software Investigated

Ubuntu is a Debian-based Linux operating system (OS) with its development led by Canonical Ltd. owned by Mark Shuttle worth. The Ubuntu distribution is free open source software, licensed under the GNU General Public License, coupled with various other free licenses. The Ubuntu OS has several versions such as desktop version, server version and the mobile version. The desktop and server versions differ in terms of the graphical user interface package used. For the desktop version, the UNITY or GNOME graphical user interface is preinstalled by default whereas the server version has a command line interface by default. The mobile version of Ubuntu is coupled with a real time kernel to handle deterministic operations such as mobile calls. The mobile version of Ubuntu is fairly new, and currently in its early stages and therefore will be omitted from this study [8] [9].

For the simplicity, this paper will focus on the primary Ubuntu distribution, and other derivatives such a Kubuntu or Xubuntu will be considered out of scope for this study.

TABLE 1 Ubuntu Versions [8] [9]

Version	Code Name	Release Date	Supported Until	
			Desktop	Server
4.10	Warty Warthog	20-10-2004	S - 30/04/2006	
5.04	Hoary Hedgehog	08-04-2005	S - 31/10/2006	
5.10	Breezy Badger	13-10-2005	S - 13/04/2007	
6.06 - LTS	Dapper Drake	01-06-2006	14/07/09	01/06/11
6.10	Edgy Eft	26-10-2006	S - 25/04/2008	
7.04	Feisty Fawn	19-04-2007	19/10/2008	
7.10	Gutsy Gibbon	18-10-2007	18/04/2009	
8.04 - LTS	Hardy Heron	24-04-2008	12/05/11	09/05/13
8.10	Intrepid Ibex	30-10-2008	30/04/2010	
9.04	Jaunty Jackalope	23-04-2009	23/10/2010	
9.10	Karmic Koala	29-10-2009	30/04/2011	
10.04 - LTS	Lucid Lynx	29-04-2010	09/05/13	04/2015
10.10	Maverick Meerkat	10-10-2010	10/04/2012	
11.04	Natty Natwhai	28-04-2011	28/10/2012	
11.10	Oneiri Ocelot	13-10-2011	09/05/2013	
12.04 - LTS	Precise Pangolin	26-04-2012	04/2017	
12.10	Quantal Quetzal	18-10-2012	04/2014	
13.04	Raing Ringtail	25-04-2013	01/2014	
13.10	Saucy Salamander	17-10-2013	04/2014	
14.04 - LTS	Trusty Tahr	17-04-2014	04/2019	

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

V. FEATURE EVOLUTION

Ubuntu 4.0 was the first version of the Ubuntu OS, released on the 26th October 2004 [8]. Every 6 months, Ubuntu released a new version except fourth and fifth version. The fourth release was designated as a Long Term Support (LTS) release with a support timeframe of five years. The support mentioned in this context refers to the OS updates and the paid technical support. The LTS releases are primarily meant for enterprise users and for large scale deployment of the operating system. The naming conventions used in each release of the Ubuntu OS are based on animal names whose appearance or habits reflects the new features of that particular release. The numeric assignment of the release versions represents the year and month of the release date [8][9].

VI. CHANGE LOG ANALYSIS

The following section shows the results of change log analysis on major changes made in every release of the Ubuntu OS. In our research, we have only focused on the Ubuntu Desktop version. The change log analysis was conducted on a single major release of each year. Corrective measure or software bug fixed applied to the OS are not been included in this section due to its vastness and as it is not expected to contribute significantly to the current study.

The 4.10 was the initial version of Ubuntu OS with limited features. The 5.10 version release focuses on improving user experience by adding more features and functionalities that enhance the usability and improvements to its graphical user interface (GUI) environment. In addition, this version attempts to broaden hardware support. The version 6.06 LTS enhance usability by introducing features such as Live CD that simplifies installation process and graphical interface for certain task such as shutdown. In version 6.10, added more features to improve the user experience and graphical interface. Version 7.04, added more the features and virtualization capabilities and dropped support for some older hardware. The version 8.04 LTS continue to serve the latest and greatest open source technologies in a high quality, easy to use Linux distribution, and focusses on improving new features. The 9.04 version improved in boot time performance and features new unsplash screen with a new login screen and also support for both Wacom hot pugging and netbook. In 11.04 versions they replaced the GNOME 2 Shell with Unity user interface as default. The 10.04 and 12.04 versions were LTS with additional features. Version 13.10 focuses on changing features to support current mobile devices standards, and provide 64-bit architecture support.

VII. CHANGE LOG AUDITS

In order to audit the software changes, the Chapin's model as shown in Figure 1 was used.

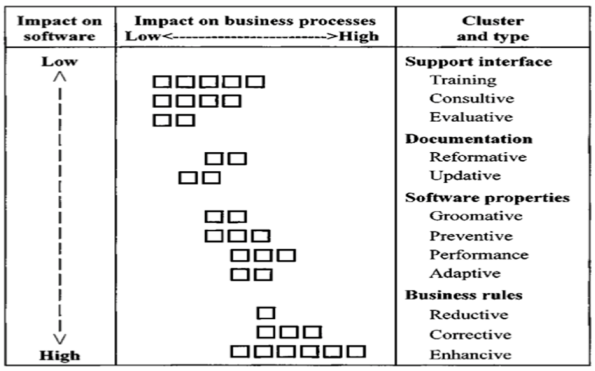


Figure 1. Chapin's model [11]

In contrast with the Swanson software evolution model, the Chapin's model provides a wider coverage of the software evolution coupled with fine grained and clearer classifications of software maintenance. The Swanson model classifies software maintenance into corrective type, perfective type, adaptive type and preventive type. On the other hand, the Chapin's model separates the software maintenance type into four clusters namely the support interface, documentation, software properties and business rules. The software maintenance types included in these clusters are listed as shown in Figure 1.

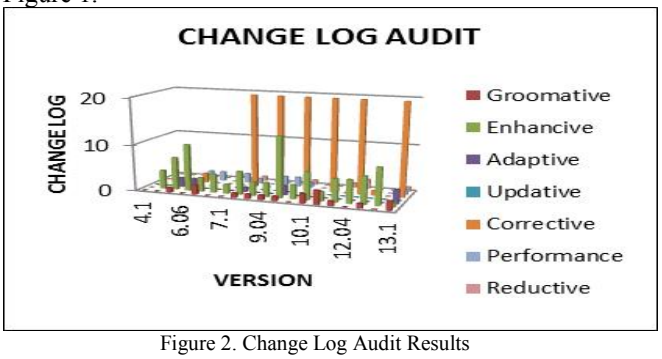


Figure 2. Change Log Audit Results

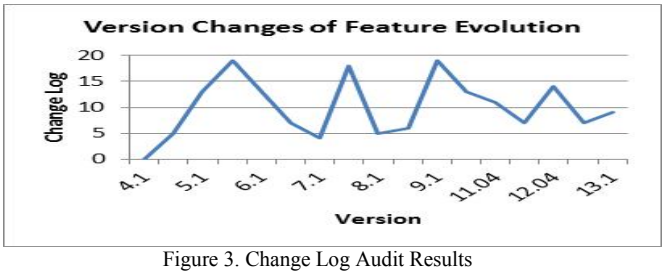


Figure 3. Change Log Audit Results

From the analysis result, we can observe that the majority of software maintenance undergone by the Ubuntu OS is of enhancive type, groomatic type and corrective type. Version 9.04, 10.1, 12.04 and 13.1 shows major changes in correctiveness of OS, while version 11.04 and Version 13.10 of Ubuntu underwent significant changes as major components of the operating system were replaced with newer components such as the replacement of the GNOME user interface with the Unity user interface and the replacement of the X11 display with the Mir display server for enhanced mobile device support.

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

The analysis process using the Chapin's model has several flaws. The effectiveness of the analysis approach is limited by the amount of published changes for the Ubuntu operating system across revisions. During the course of this study, change logs for the Ubuntu OS were found to be poorly maintained with a significant inconsistency on the types of information published.

VIII. Growth Analysis

To analysis the software growth of the Ubuntu operating system, the Lehman's Laws of software evolution was employed. In this context, an E-type system is a software system developed to perform real-world activities.

In order to perform the source code metrics analysis, the open source UCC (Unified Code Count) application is used to automate this process [12]. For a broader understanding of the software growth and evolution of the Ubuntu OS, the source code metrics analysis is performed on Ubuntu 5.04, Ubuntu 6.06, Ubuntu 8.04, Ubuntu 10.04, Ubuntu 12.10 and Ubuntu 13.10. Three types of metrics are gathered throughout the analysis namely the number of classes (files in this context), number of functions or methods and the number of lines of codes. Empty lines, source code comments and headers will be ignored in the metrics analysis to ensure only functional codes are considered.

A. Metrics against Age Analysis

The plot of the source code metrics against the software age depicts the overall growth of the software through time. Figure 2 shows the interpolated software growth and changes based on the analysis of five versions of Ubuntu.

B. Metrics against Release Revisions

The plot of the source code metrics against the software age depicts the overall growth of the software across versions which are Ubuntu 5.04, Ubuntu 6.06, Ubuntu 8.04, Ubuntu 10.04, Ubuntu 12.10 and Ubuntu 13.10.

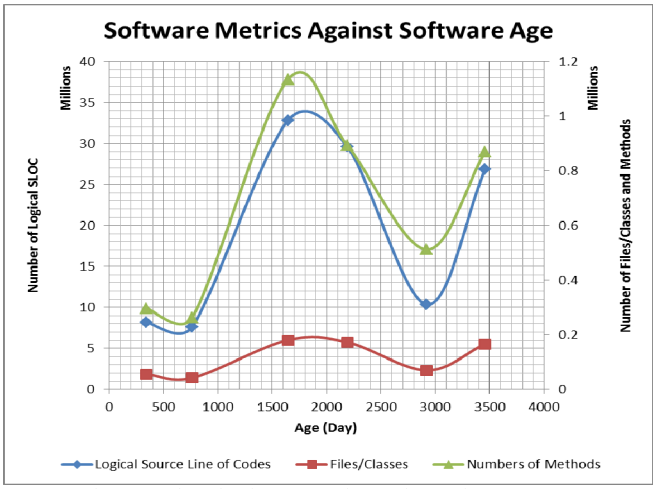


Figure 2. Plot of software metrics against software age

C. Relative Metric against Age Analysis

The plot of the normalised relative changes against the software age depicts the relative changes made to the software through time. Figure 4 shows the interpolated relative software changes based on the analysis of five versions of Ubuntu.

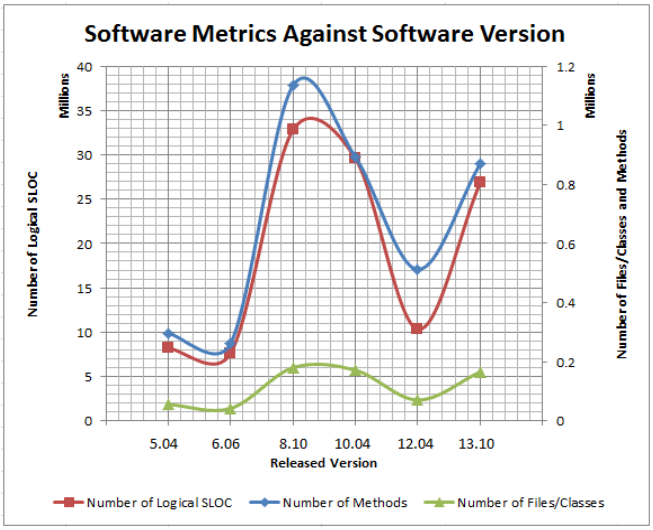


Figure 3. Plot of software metrics against software version

D. Relative Metric against Release Revision

The plot of the normalised relative changes against the software age depicts the relative changes made to the software across versions.

From Figure 2 and Figure 3, it can be observed that the software metrics peak for the release of Ubuntu 8.10 followed by a gradual decrease. Ubuntu 13.10 is the turning point of the software growth in which software metrics shows significant increases. The software growth and evolution of the Ubuntu operating system trend can be observed from the relative plots shown in Figure 4 and Figure 5.

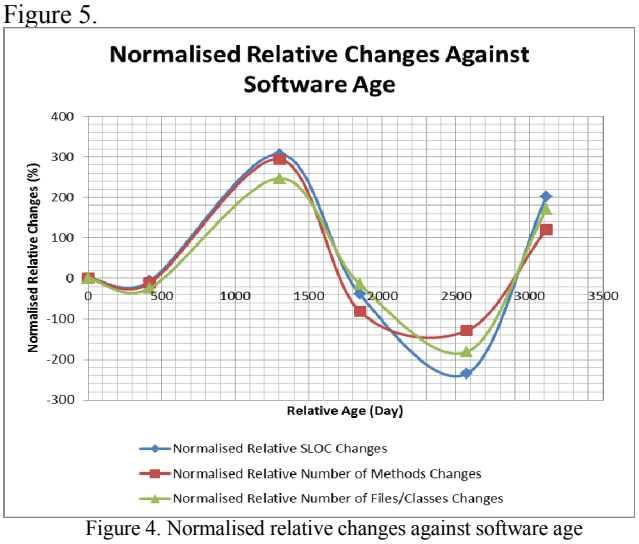


Figure 4. Normalised relative changes against software age

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

The significant decrease in software metrics from Ubuntu 8.10 to Ubuntu 10.04 may be due to the obsolescence of support for Intel's LPIA (Low Power Intel Architecture). The proposed decision by Canonical to drop support for LPIA architecture, a microarchitecture employed by Intel's Atom series processor, corresponds with Lehman's law of increasing complexity. The decision to drop support is to facilitate in developing a simpler kernel with LPIA optimization using an i386 spin-off [13].

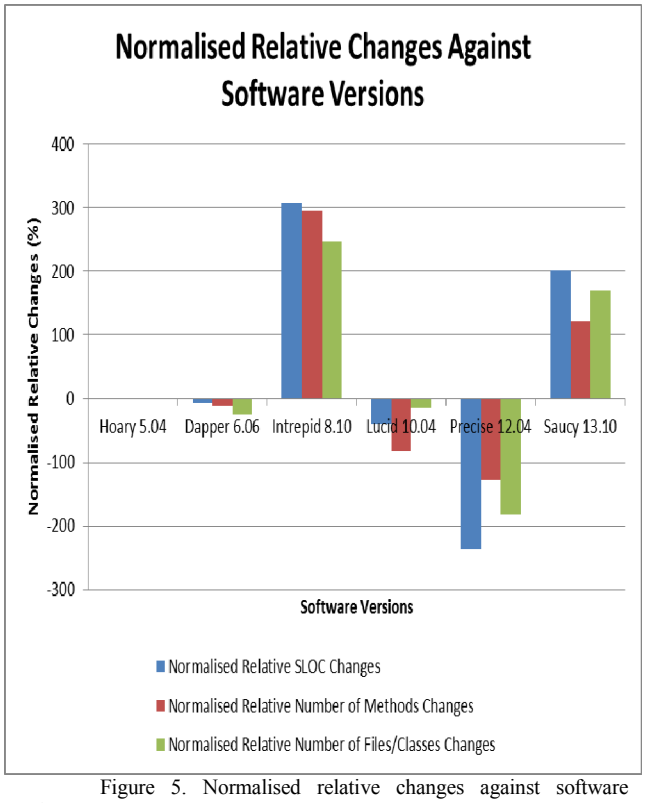


Figure 5. Normalised relative changes against software versions

The overall growth and evolution of the Ubuntu operating system abides by the Lehman's law of software self-regulation. By observing Figure 4, the software relative growth trend can be statistically represented using geometric curve fitting based regression analysis.

In addition to that, the development process of the Ubuntu also abides by the Lehman's law of continuing growth and changes. From Figure 2, it can be observed that Ubuntu is continuously adapting to newer technologies by removing obsolete components and adding more efficient components to the system such as the replacement of the GNOME user interface with the "Touch Friendlier" Unity user interface.

On the other hand, Ubuntu's software growth loosely abides by the Lehman's law of conservation of familiarity and the conservation of organizational stability. By observing Figure 4 and Figure 5, the incremental change and average activity rate varies broadly and is highly dependent on the Canonicals' headings and roadmaps.

IX. Change Analysis

To perform a change analysis on the Ubuntu OS, the Gini coefficient is used to provide a measure of statistical dispersion to analyze how changes are made and distribution of changes or source codes across the specified Ubuntu revisions.

$$Gini\ Coefficient = 1 - 2 * Lorenz\ Curve\ Area \quad \text{(eq1)}$$

$$Lorenz\ \ Curve\ \ Area = \int_0^1 y\ dx \quad \text{(eq2)}$$

where, y is the regression line.

To facilitate in the calculation of the the Gini coefficient, the Matlab mathematical software will be used. The Gini coefficient for the following section will be calculated using the Matlab algorithm as followed:

$$y = c_6 x^6 - c_5 x^5 + c_4 x^4 - c_3 x^3 + c_2 x^2 - c_1 x + c_0 \quad \text{(eq3)}$$

In the section, a non-linear regression line will be used to model the plotted Lorenz curve specifically a 6th order polynomial regression line (eq3). The Gini coefficient is determined by the integration of the polynomial equation followed by the insertion of the obtained value into eq1.

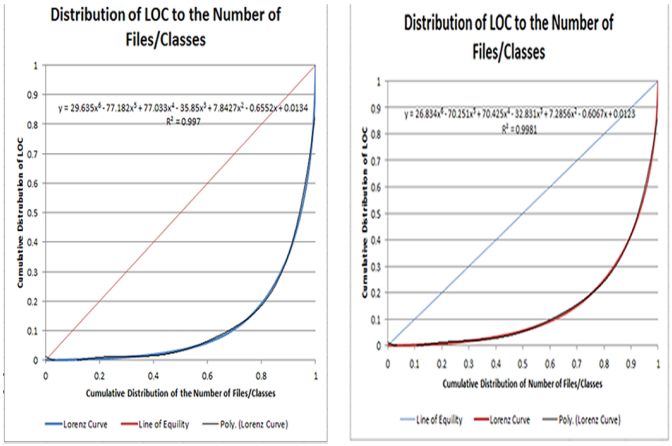
A. Logical LOC to Classes/Files Distribution

In the section, change analysis will be performed on six versions of the Ubuntu operating system namely Ubuntu 5.04, Ubuntu, Ubuntu 6.06, Ubuntu 8.10, Ubuntu 10.04, Ubuntu 12.04 and Ubuntu 13.10.

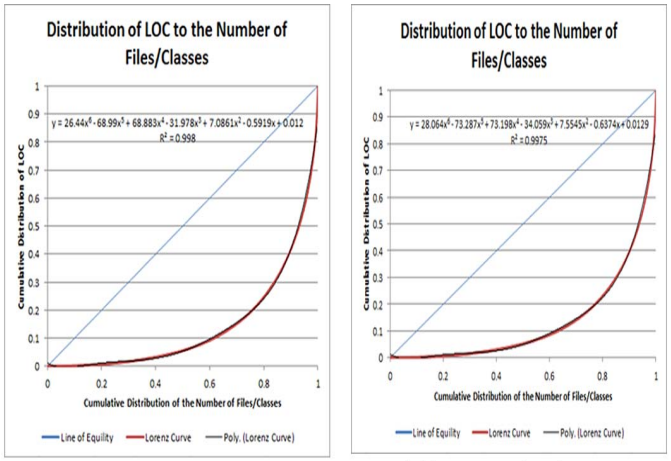
TABLE 2 Lorenz Curve

Ubuntu	Lorenz Curve Properties Regression Line Equation	Gini Coefficient
5.04	$y = 29.635x^6 - 77.182x^5 + 77.033x^4 - 35.85x^3 + 7.8427x^2 - 0.6552x + 0.0134$	0.7719
6.06	$y = 26.834x^6 - 70.251x^5 + 70.425x^4 - 32.831x^3 + 7.2856x^2 - 0.6067x + 0.0123$	0.7207
8.10	$y = 26.44x^6 - 68.99x^5 + 68.883x^4 - 31.978x^3 + 7.0891x^2 - 0.5919x + 0.012$	0.7200
10.04	$y = 28.064x^6 - 73.287x^5 + 73.198x^4 - 34.059x^3 + 7.5545x^2 - 0.6374x + 0.0129$	0.7363
12.04	$y = 30.251x^6 - 79.014x^5 + 78.851x^4 - 36.719x + 8.1262x^2 - 0.6945x + 0.0139$	0.7632
13.10	$y = 29.898x^6 - 75.454x^5 + 75.279x^4 - 34.968x^3 + 7.7216x^2 - 0.6516x + 0.0131$	0.4591

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

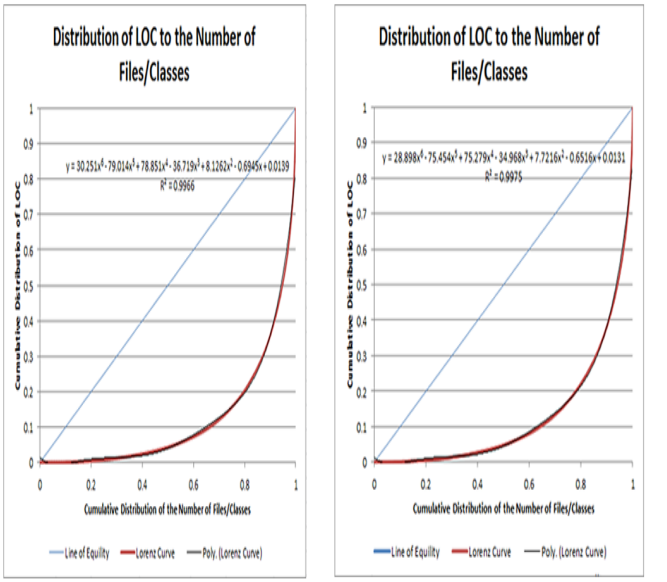


Distribution of LOC to the Number of Files/Classes (Ubuntu 5.04) Distribution of LOC to the Number of Files/Classes (Ubuntu 6.06)



Distribution of LOC to the Number of Files/Classes (Ubuntu 8.10) Distribution of LOC to the Number of Files/Classes (Ubuntu 10.04)

Figure 6. LOC Ubuntu 5.04, 6.06, 8.10 & 10.04



Distribution of LOC to the Number of Files/Classes (Ubuntu 12.04) Distribution of LOC to the Number of Files/Classes (Ubuntu 13.10)

Figure 5. LOC Ubuntu 12.04 & 13.10

B. *Gini Coefficient Analysis*

Figure 8 Gini Coefficient of the SLOC to Files/Classes Distribution

From Figure 8, we can observe that the distribution of software measure to the files/classes is constantly changing across different revisions of Ubuntu. A notable drop in the Gini coefficient is observed from Ubuntu 13.10 which suggests software architectural changes that lead to a more uniformly distributed SLOC to files/classes ratio. This may be in effort to reduce the overall complexity of the software.

X. RESULTS DISCUSSION

Based on the analysis presented in the previous section of this study, the Ubuntu OS has been continuously evolving and growing in order to achieve both functional and non-functional requirements.

Using the Chapin's model to perform the software change log audit, it can be observed that the changes are primarily focused on the software properties clusters and the business rules clusters. These clusters includes maintenance types such the groomatic type, preventive type, performance type, adaptive type, reductive type, corrective type and the enhancive type. These software maintenance types allow the Ubuntu OS to constantly improve and enrich the functionalities provided and improve the overall performance. In addition to that, these software maintenance types allow Ubuntu to cope with the hardware environment changes as well as changes the user expectations. However, the change log audit also shows that the support interface clusters and the documentation clusters are poorly maintained. These clusters focus on improving the software documentation for both users and developers. The substandard realization of these types of software maintenance can be observed from the web based user documentation in which obsolete information are not updated.

From the software growth analysis, the Ubuntu OS has undergone a significant amount of changes. From the analysis result, the growth rates across different versions of Ubuntu are not consistent. Positive growth rate can be

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.

observed from Ubuntu 5.04 to Ubuntu 8.10 and from Ubuntu 12.04 to Ubuntu 13.10 while negative growths are observed from Ubuntu 8.10 to Ubuntu 12.10 as shown in Figure 3, Figure 3, Figure 4 and Figure 5. Negative growth in software is triggered by the deprecation or obsolescence of major components of Ubuntu such as the user interface component or the hardware supports. Overall, the Ubuntu operating system has a positive software growth.

From the software change analysis, the distribution of SLOC to the files/classes is skewed. This leads to the inference that the complexity of the files/classes differs significantly from one another. However, Ubuntu 13.10 shows a significant drop in the Gini coefficient which suggest that efforts are being made to curb the complexity issue of the software architecture.

XI. LIMITATIONS

The method of analysis perform in this study has several limitations. Change log auditing assumes that the change log for Ubuntu is properly documented. Unfortunately, the change logs for Ubuntu are maintained by a community of volunteer software developers. These change logs are documented without any specified documentation standards and may have irrelevant or omitted information.

In addition to that, the variation of software metrics used in this study is limited to those presented by the UCC Code Count application. Ubuntu is software that is developed using multiple programming languages, a more sophisticated software metrics is required to perform an accurate analysis of the software growth. The logical source line of codes (SLOC) metric used in this analysis shows only the software growth in size. It is not the appropriate means of analyzing the software growth in terms of the functional and non-functional requirements. Besides that, the representation of a file as an object oriented class in this study assumes that the Ubuntu operating system is structured in the appropriate manner for this assumption to be valid.

This, as an initial study, did not explore the entire depth of analysis on Ubuntu. Thus, the analysis results presented in this paper may or may not reflect the precise actual growth and evolution of the software. Analysis on the software growth is only performed on Ubuntu versions 5.04, 6.06, 8.10, 10.04, 12.04 and 13.10.

XII. CONCULSION

In conclusion, the growth analysis and the change analysis conducted on the Ubuntu operating system shows that it is undergoing continuous changes to its functional and non-functional requirements. The software growth cycle of the Ubuntu operating system is observed to be in accordance to the Lehman's law of software evolution. However, the validity of this software growth and evolution paper is constrained by the loss of in information due to the poorly maintained documentation for Ubuntu.

ACKNOWLEDGMENT

We acknowledge the contributions of Jason Yong, Ting Kuok How and Kuan Yu Shen in assisting to perform the analysis.

REFERENCES

[1] Vasa, Rajesh. "Growth and change dynamics in open source software systems." Faculty of Information and Communication Technologies (2010): 254.

[2] T. DeMarco. Controlling Software Projects: Management, Measurement and Estimation. Yourdon Press New York, 1982.

[3] Vasa, R., Lumpe, M., Branch, P., & Nierstrasz, O. (2009, September). Comparative analysis of evolving software systems using the Gini coefficient. InSoftware Maintenance, 2009. ICSM 2009. IEEE International Conference on(pp. 179-188). IEEE.

[4] M. M. Lehman. Programs, Life Cycles, and Laws of Software Evolution. Proceedings of the IEEE, Special Issue on Software Evolution, 68(9):1060–1076, Sept. 1980.

[5] E. Swanson. The Dimensions of Maintenance. In Proceedings of the 2nd International Conference on Software Engineering, pages 492–497, 1976.

[6] Karch, Erich. "Lehman's Laws of Software Evolution and the Staged-Model." http://blogs.msdn.com/b/karchworld_identity/archive/2011/04/01/lehman-s-laws-of-software-evolution-and-the-staged-model.aspx

[7] Hall, Richard S., Dennis M. Heimbigner, and Alexander L. Wolf. "Evaluating software deployment languages and schema: an experience report." Software Maintenance, 1998. Proceedings. International Conference on. IEEE, 1998.

[8] CJohnston, "Ubuntu Wiki Releases," Canonical, 22 10 2013. [Online]. Available: https://wiki.ubuntu.com/Releases. [Accessed 9 11 2013].

[9] Francesco-infante, "Realease Naming Scheme," Canonical, 21 10 2013. [Online]. Available: https://wiki.ubuntu.com/DevelopmentCodeNames.

[10] Francesco-infante, "Realease Naming Scheme," Canonical, 21 10 2013. [Online]. Available: https://wiki.ubuntu.com/DevelopmentCodeNames. [Accessed 9 11 2013].

[11] J. F.-R. D. P. Nazim H. Madhavji, "Software Evolution and Feedback: Theory and Practice," in Software Evolution and Feedback: Theory and Practice, John Wiley & Sons, 2006, p. 612.

[12] http://sunset.usc.edu/research/CODECOUNT/

[13] M. Larabel, "Ubuntu To Stop Supporting LPIA Architecture," Phoronix, 24 November 2009. [Online]. Available: http://www.phoronix.com/scan.php?page=news_item&px=NzczOA. [Accessed 12 November 2013].

[14] K. Noyes, "10 reasons to choose Ubuntu 12.10 over Windows 8," 2012. [Online]. Available: http://www.pcworld.com/article/2013431/10-reasons-to-choose-ubuntu-12-10-over-windows-8.html?page=2.

[15] Diffen, "Linux vs Windows," [Online]. Available: http://www.diffen.com/difference/Linux_vs_Windows.

[16] U. T. Board, "Bugs: Ubuntu," Canonical, [Online]. Available: https://bugs.launchpad.net/ubuntu.

[17] Vorlon, "Hardy Release Notes," Canonical, 28 1 2010. [Online]. Available: https://wiki.ubuntu.com/HardyReleaseNotes.

[18] Linuxfx, "Linuxfx Computing," 2012. [Online]. Available: http://linuxfx.sdf.org/ubuntu.html.

[19] Brad-figg, "Ubuntu Lucid 10.04," Canonical, 16 2 2012. [Online]. Available: https://wiki.ubuntu.com/LucidLynx/ReleaseNotes/ChangeSummary/10.04.4.

Authorized licensed use limited to: University of the Philippines - Los Banos. Downloaded on April 29,2024 at 20:23:37 UTC from IEEE Xplore. Restrictions apply.