An Empirical Comparison of the Development History of CloudStack and Eucalyptus

Ahmed Zerouali
Software Engineering Lab
University of Mons - Belgium
ahmed.zerouali@umons.ac.be

Tom Mens
Software Engineering Lab
University of Mons - Belgium
tom.mens@umons.ac.be

ABSTRACT

Open source cloud computing solutions, such as *CloudStack* and *Eucalyptus*, have become increasingly popular in recent years. Despite this popularity, a better understanding of the factors influencing user adoption is still under active research. For example, increased project agility may lead to solutions that remain competitive in a rapidly evolving market, while keeping the software quality under control. Like any software system that is subject to frequent evolution, cloud computing solutions are subject to errors and quality problems, which may affect user experience and require frequent bug fixes. While prior comparisons of cloud platforms have focused most often on their provided services and functionalities, the current paper provides an empirical comparison of *CloudStack* and *Eucalyptus*, focusing on quality-related software development aspects. More specifically, we study the change history of the source code and its unit tests, as well as the history of bugs in the *Jira* issue tracker. We found that *CloudStack* has a high and more rapidly increasing test coverage than *Eucalyptus*. *CloudStack* contributors are more likely to participate in development and testing. We also observed differences between both projects pertaining to the bug life cycle and bug fixing time.

CCS CONCEPTS

• **Software and its engineering** → *Software testing and debugging*; *Software defect analysis*; • **General and reference** → *Empirical studies*;

KEYWORDS

Software Analysis, Cloud Computing, Open Source, Empirical Analysis, Unit Testing, Bug reports

ACM Reference format:
Ahmed Zerouali and Tom Mens. 2017. An Empirical Comparison of the Development History of CloudStack and Eucalyptus. In *Proceedings of ICSDE '17, Rabat, Morocco, July 21–23, 2017,* 6 pages.
https://doi.org/10.1145/3128128.3128146

ICSDE '17, July 21–23, 2017, Rabat, Morocco

ACM ISBN 978-1-4503-5281-9/17/07…$15.00
https://doi.org/10.1145/3128128.3128146

1 INTRODUCTION

Due to market pressure, the use of open source solutions to deploy cloud computing as a new service concept has grown dramatically in recent times, which made it gain a significant attention from industry and researchers. This service paradigm provides virtualized computing resources as a service on demand, and allows companies focus on their business issues rather than conceiving and managing complex infrastructures. In order to improve service quality infrastructure planning, and selection of the right software solutions, users and cloud creators need to evaluate and compare the performance of the features offered by competing platforms.

Trust is the main challenge of open source and commercial cloud computing solutions. Cloud users need assurance about the availability and maintainability of the solution they are adopting, while cloud creators need to understand how their competitors are maintaining and improving their product.

This article focuses on the quality aspects of cloud computing solutions in *Java*. Like any other software system, cloud solutions are subject to frequent evolution, and may suffer from bugs and other quality problems that may affect the user experience. An historical analysis and comparison of how competing cloud solutions are developed and maintained over time is therefore useful for their creators and users. For this reason, we study and compare two popular Java-based open source cloud computing systems: *CloudStack* and *Eucalyptus*. Other popular open source cloud computing solutions such as *OpenStack* and *OpenNebula* are not considered in this study. As they are not developed in *Java*, comparing their software development history with *Java*-based solution would be unfair and not meaningful.

Lehman's *laws of software evolution* [9] describe forces driving new development and forces that slow down this progress. The most popular of these laws are "Continuing Change", "Continuing Growth" and "Declining Quality". Based on these laws we identified three research questions:
RQ1: How do open source cloud systems grow and evolve?
RQ2: How thoroughly are cloud systems tested and how does this evolve over time?
RQ3: How do cloud system contributors manage their issue and bug reports?

2 RELATED WORK

Many studies have been conducted to evaluate the performance improvement of open source cloud solutions such as *Eucalyptus*[1] [11], *CloudStack*[2] [8] and *OpenStack*[3] [14]. As discussed below, there have been many studies about open source cloud computing solutions that enable to set up private and hybrid clouds, with as main focus the analysis and comparison of middleware platform features, architectures and performance.

Al-Mukhtar et al. [1] evaluated the performance of *Eucalyptus* and *CloudStack* cloud virtual machines covering versatile parameters including the performance of the cloud management platform. Performance of VMs in term of CPU usage, memory bandwidth, disk I/O speed and networking performance is rated as key points in their evaluation. Chilipirea et al. [4] and Vogel et al. [15] performed a similar comparative analysis for the *OpenStack*, *OpenNebula* and *CloudStack* platforms.

To date, and to our knowledge, there has been no study that has compared the technical development history of the available open source cloud computing solutions. However, few researchers have already analysed bug reports in open source cloud systems. Jenkins et al. [6] suggested an intelligent framework for testing cloud platforms and infrastructures and they demonstrated its applicability on a prototype framework for testing the Google App Engine. Frattini et al. [5] presented an empirical analysis tailored to open source clouds in which they studied 146 bug reports from Apache Virtual computing. Empirical bug analysis has been demonstrated to be beneficial for several software systems, such as desktop [12], mobile operating systems [10] and mobile apps [2]. Moreover, many studies have shown how the analysis of bug discovery over time can be useful to predict the residual number of bugs in the code [13, 16].

In contrast to these related works, this paper compares the development and testing history of two open source cloud computing solutions for Java and analyses their issue reports.

3 METHODOLOGY

To determine the most appropriate cloud solutions for our analysis we searched for the most popular open source projects that use the same programming language for their development, that use the same issue and bug tracker, and that are hosted on the same version control system. We selected the most competitive *Java*-based open source cloud computing projects that are using the *Jira* issue for their bug and issue tracking and that are hosted on *GitHub*: *CloudStack* and *Eucalyptus*.

To enable historical analysis of the co-evolution of the source code and unit test code in the considered projects, we created a local clone of their *GitHub* repository. We extracted the added, removed and modified *Java* files as well as the number of committers that participate in the development in each commit since the creation of the project. In order to analyse the issue reports of the considered projects, we extracted all available information that is reported in *Jira*, a commercial software product that provides bug tracking, issue tracking, and project management functions.

[1]https://www.eucalyptus.com/wiki
[2]https://cloudstack.apache.org/
[3]https://www.openstack.org/

Source code is the main artefact affecting the quality of a software project throughout its development history. However, many other technical artefacts should be considered when producing high quality software. Software testing (e.g., integration testing, unit testing, functional testing, ...) is a generally acknowledged way to increase the quality of software.

To evaluate unit test coverage in each project, for each month, we analysed the first snapshot of each considered project, by looking at the import statements in each *Java* file of the project in order to identify the ones in which we can find the usage of popular unit testing libraries such as *JUnit*, *TestNG* and *Spring* [17]. To identify which *Java* classes are targeted by the unit test cases we parsed the test *Java* files and extracted the classes that some or all of their methods have been tested.

A descriptive summary of the two considered *Java* cloud projects is provided in Table 1. The last considered commit of both projects in our study was 30 March 2017.

Table 1: Descriptive statistics about Java-based cloud solutions *CloudStack* and *Eucalyptus*.

	CloudStack	*Eucalyptus*
First commit	2010-08-11	2009-01-06
Contributors	241	45
Analysed issues	9,773	15,822
Commits	30,466	26,225

4 EMPIRICAL EVALUATION

We address each research question in a separate subsection by means of tables, visualisations and statistical tests.

4.1 How do open source cloud projects grow and evolve?

To answer this question, we analysed the commit history on *GitHub* for both considered projects since their first commit and until 30 March 2017.

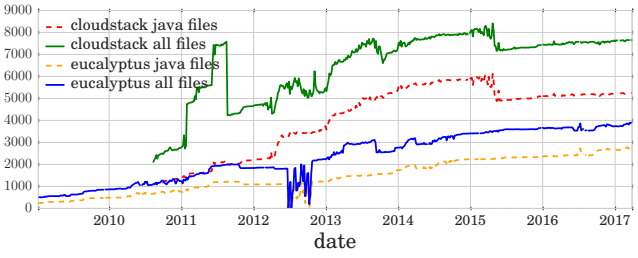

Figure 1: Growth evolution of cloud solutions

Figure 1 shows the evolution of the size of both projects in terms of total number of files and total number of *Java* files. *CloudStack* appears to be twice as big as *Eucalyptus*, even if the latter is older than the first. *Eucalyptus* started with a limited number of files, while *CloudStack* started with a big number of files since the first

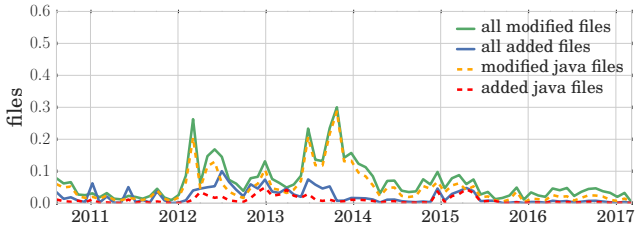

Figure 2: Proportional number of added and modified files in *CloudStack* at each point in time.

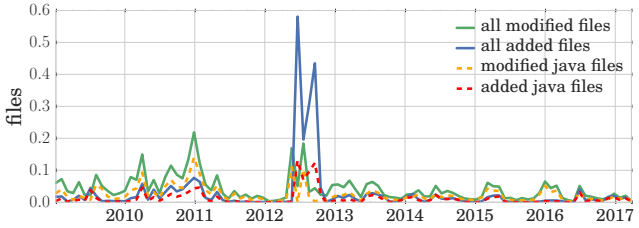

Figure 3: Proportional number of added and modified files in *Eucalyptus* at each point in time

commit, which can be explained by the fact that *CloudStack* was already in development before coming to *GitHub* in August 2010. Indeed, its *GPLv3* version was released in May 2010. *Eucalyptus* was split into two editions, open-core and open source. But after *July 2011*, when *Citrix* purchased *Cloud.com*, the number of files of the *CloudStack* system decreased, to start increasing again after the donation of the project to the Apache Software Foundation in April 2012.

However, *Java* files are the main artefact affecting the growth of both projects. For both systems, the number of *Java* files started with 50% of all files, and ended up to 70%. Furthermore, we examined the growth evolution curve trend for *Java* files, and we found that the trendline of the linear regression model was a good fit for the project *Eucalyptus* with a coefficient of determination $R^2 = 0.92$, where *CloudStack*'s trendline was superlinear fitting the exponential model with a coefficient of determination $R^2 = 0.85$.

To understand how active is the community of both projects, we calculated the number of added and changed files of any type, and the *Java* files in particular, as well as the number of developers participating in development in each month.

Figure 2 and Figure 3 show the proportional number of the added and modified files, including *Java* files, to the size of both projects at each point in time. For both systems we found that the edited files in most times, are files related to programming (i.e., *Java* files). However, we noticed that some unusual but expected activities happened after some events, related to the project management.

For example, when *Eucalyptus* returned to be fully open source in *June 2012*, a large number of files was added to the project, and after *CloudStack* became a *Top-Level Project* of Apache in March 2013 and released the first stable version after graduation, a high number of files, mostly *Java* files, was edited.

In the other periods, where no special events happened, the percentage of the edited files was between 1% and 10% of all files in both *CloudStack* and *Eucalyptus*, while the percentage of the added files was between 0% and 6%.

For each month, we calculated the number of contributors participating in the development of the considered projects. We found that the mean number of contributors participating in *CloudStack* development was twice as high as in *Eucalyptus*, which can be explained by the obvious difference in the total number of contributors between the two systems. For both projects, the highest numbers of contributors were found in the period between 2012 and 2014.

4.2 How thoroughly are cloud systems tested and how does this evolve over time?

In order to give an answer to such question, we focused on unit tests only, and compared the presence and coverage of *Java* unit tests in *CloudStack* and *Eucalyptus*. Both systems make use of software libraries that support unit testing. While *CloudStack* uses the frameworks *Spring*, *TestNG* and *JUnit*, *Eucalyptus* only uses *JUnit*. To calculate the number of tested *Java* classes, for each *Java* file, excluding *Java* files that contain test cases, we extracted the *Java* classes inside and we analysed them.

The percentage of *Java* classes that are tested in each considered system is shown in Figure 4. We observe that the number of tested *Java* classes in *CloudStack* is increasing over time. Until 2013, only 10% of *Java* classes were tested, but after the release of the first stable version, this number changed consecutively to achieve 74% in 2017. However, the change of the number of *Java* files that contain test cases was not important. In 2013, 1.7% of all *Java* files in *CloudStack* were test files, this number slightly changed to become 10% in 2017. For *Eucalyptus*, only a small portion of *Java* classes were tested across the system lifetime, the maximum was in December 2016 with 4% of all classes. However, the number of test files co-evolved closely with the number of tested *Java* classes, the maximum was in December 2016 with 2% of all *Java* files. These results reflect that *CloudStack* is following a good test-driven development approach.

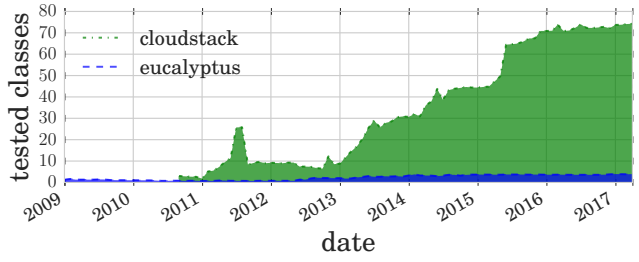

Figure 4: Percentage of *Java* classes being tested in both cloud solutions.

In order to know how many developers participated in unit testing development, for each year we calculated for each *Java* test file (i.e., each file containing JUnit test cases) the number of developers that participated in its development. We observe that the

number of *CloudStack* developers participating in test code is high, the maximum number was in 2013 with 65 different developers. For *Eucalyptus*, only a maximum of 8 different developers worked on *Java* test files in 2014.

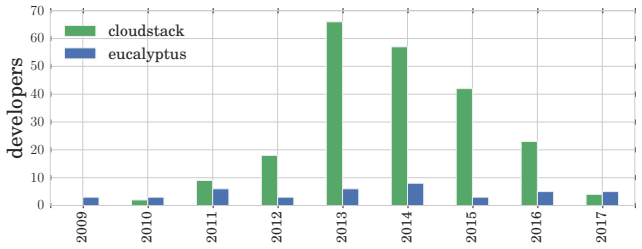

Figure 5: Number of *Java* test files touched by a specific number of developers.

4.3 How do cloud system contributors manage their issue and bug reports?

To answer our third research question, we analysed the publicly available issue reports of *CloudStack* and *Eucalyptus* found on the *Jira* issue tracker[4] since 2012. Reports can be created and managed by whoever uses the platform (e.g., developers, administrators, consumers). Developers can report a bug during any stage of the software life cycle (e.g., development, testing, ...).

4.3.1 Issue Report Analysis. Figure 6 shows pie charts exhibiting the different issue priorities and their percentage in both considered projects. In *Eucalyptus*, 72% of all issues have a "Major" priority. 49% of these issues and 58% of all issues are bugs. With 55%, the percentage of the "Major" issues in *CloudStack* is less than in its competitor, but the percentage of bugs of this proportion is higher than in *Eucalyptus* with 67%, and 76% of all issues. For the other priorities, for both projects, the percentage of bugs is higher than 70% (higher than 85% for "Blocker" and "Trivial" priorities).

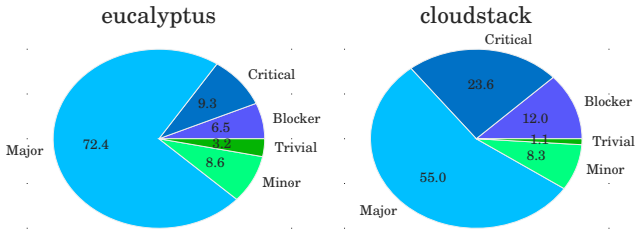

Figure 6: Percentages of issue priorities found across both cloud projects

Figure 7 shows, for each issue type that is common between *CloudStack* and *Eucalyptus*, a violin plot that compares between the considered projects, the distribution of the time between the creation date and the last update of a closed issue across all found issues. For issues related to new features and improvements, we did not find a significant difference between the time that *CloudStack* and *Eucalyptus* issues take before being closed.

The bug issues in *CloudStack* take less time before closing them than the bugs in *Eucalyptus*. To confirm our observations, we used a one-sided *Mann-Whitney U* test to verify if there is a statistical significance difference in the number of days between the creation date and the closing date. We found a statistical significance at p-value<0.01 between the two projects, where *Eucalyptus* bugs take more days before closing them compared to the bugs of *CloudStack*. We also found that, contrary to *Eucalyptus*, *CloudStack* has an issue type related to "Tests" only.

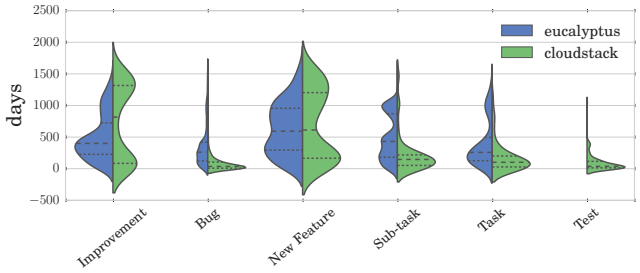

Figure 7: Distribution of the number of days between creation date and last update of each reported issue, classified by type.

4.3.2 Bug Report Analysis. Figure 8 shows the evolution of the number of bugs created over time for both *Eucalyptus* and *CloudStack*. We observe that for both projects, the number of bugs reported over time tends to decrease. The highest number of bugs was found in 2013 for *CloudStack* and in early 2014 for *Eucalyptus*. The number of bugs decreased faster in *CloudStack*, from 600 created bugs per month in 2013 to 50 bugs per month in 2016. This can perhaps be explained by the testing approach that *CloudStack* has pursued after 2013 (cf. Section 4.2).

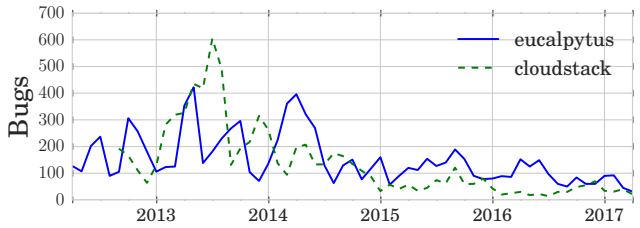

Figure 8: Number of bugs found over time

For each bug report, we extracted and calculated the number of participants. Figure 9 presents the distribution per year, and appears to reveal that the number of participants in bug handling for *Eucalyptus* is higher than in *CloudStack*. We statistically tested this hypothesis with a one-sided *Mann-Whitney U* test and confirmed it with statistical significance (p-value<0.01).

[4]https://www.atlassian.com/software/jira

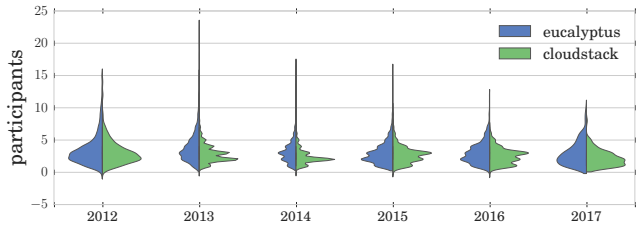

Figure 9: Number of participants in bug reports

4.3.3 Bug Life Cycle Analysis. The bug life cycle is a cyclic process followed by reported bugs throughout their lifetime. It begins when one of the contributors (e.g., tester, developer, ...) reports the bug and ends when the bug is closed, typically after a thorough verification to ensure that the bug is not reproduced. Hence, the bug report has different states in its life cycle.

In order to analyze the bug life cycle for both considered projects in this study, for each bug we extracted the history of all state transitions of the bug report. Using the *Disco* process mining tool[5] we generated a generic bug life cycle with the mean time that a bug takes to remain in each state, considering the most repeated life cycles of 9,057 bugs in *Eucalyptus* and 6,235 bugs in *CloudStack*.

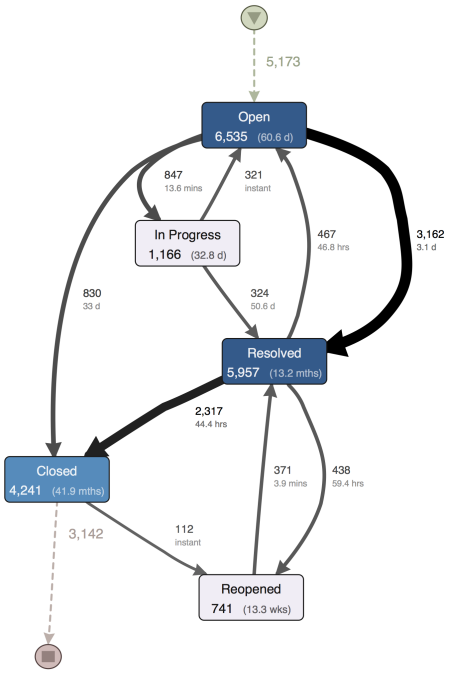

Figure 10: Bug life cycle of *CloudStack*.

We found that both cloud systems have different bug life cycles even if they use the same issue tracker. As shown in Figure 10, *CloudStack* has a generic bug life cycle with a few number of states. Most bugs in *CloudStack* follow the transition sequence: **Open → Resolved → Closed**.

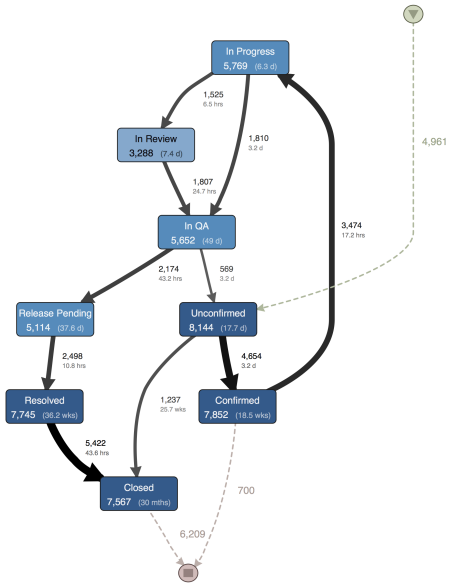

Figure 11: Bug life cycle of *Eucalyptus*

As shown in Figure 11, a typical bug life cycle in *Eucalyptus* adopts the following sequence:
Unconfirmed → Confirmed → In Progress → [*In Review*] **→ In QA → Release Pending → Resolved → Closed**.
This longer life cycle might explain why it takes longer, on average, for bug reports in *Eucalyptus* to reach the **Closed** status. It also implies that the number of participants involved in a bug fix in *Eucalyptus* tends to be higher than in *CloudStack*. While 92% of the initially **Unconfirmed** bugs in *Eucalyptus* end up in the **Closed** status, only 64% of the **Opened** bugs in *CloudStack* end up in the **Closed** status. Nevertheless, the percentage of bugs that reach the **Resolved** status is almost the same in both projects, 91% in *CloudStack* and 95% in *Eucalyptus*. A possible reason is that many *CloudStack* contributors consider the **Resolved** status as an endpoint.

However, the number of bugs that were **Reopened** after being **Resolved** or **Closed** in *CloudStack* is higher than in *Eucalyptus*, which means that in order to resolve a bug and despite the effort that this process can take, the long bug life cycle in *Eucalyptus* is a better approach to follow.

5 THREATS TO VALIDITY

Our research suffers from the same threats as other research relying on *Git* [3] and *GitHub* [7].
Our results may not be generalisable to non-*Java* cloud solutions or to closed-source industrial cloud solutions that are typically subject to more restricted development rules. While we analysed the development of two *Java*-based cloud solutions, the proposed methodology is applicable to other *Java*-based open source projects. In our approach we assume that a *Java* class is being tested if at

[5] https://fluxicon.com/disco/

least one of its methods is tested, which may lead to false positives, since a *Java* class can have many methods that together provide the functionality offered by this class. We also assume that there is a strict separation between test code and production code, i.e., the test files do not contain production code.

6 CONCLUSION AND FUTURE WORK

We analysed two popular and competitive open source cloud computing solutions developed in *Java*, namely *CloudStack* and *Eucalyptus*, based on data about their development history extracted from the *GitHub* code versioning tool and its associated *Jira* issue tracker. We studied three research questions related to the development history of the considered cloud computing systems.

We observed that, initially, *CloudStack* was growing differently, but in the last two years it started growing at the same rate as *Eucalyptus*. We observed that events related to how both projects are managed appear to affect their evolution of change and growth. We also found that *CloudStack* has a good testing approach with a better unit testing coverage than *Eucalyptus*. This approach helped to reduce the number of bugs found over time. We also found that the studied systems have different bug life cycles and take different times to resolve and close a bug.

This analysis identifies when bugs are likely to be found in future development of the cloud systems, the phases of the life cycle during which such bugs may be resolved, closed or ignored, as well as hints about the effort required to maintain and improve these systems. These findings are potentially useful to the developer communities responsible for these cloud solutions for improving development and testing activities, as well as for researchers and new contributors so they can learn from past experiences.

In future work, we will extend our comparison to include more systems like *OpenStack* and *OpenNebula*, that use different programming languages like *Python* and *C++*, and we will also take into consideration the performance and feature aspects.

7 ACKNOWLEDGMENT

This research is part of FRFC research projects T.0022.13 and J.0023.16 financed by F.R.S.-FNRS, Belgium.

REFERENCES

[1] M. AL-Mukhtar and A. A. A. Mardan. Performance evaluation of private clouds Eucalyptus versus CloudStack. *Int'l J. Advanced Computer Science and Applications*, 5(5):108–117, 2014.

[2] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru. An empirical analysis of bug reports and bug fixing in open source Android apps. In *European Conf. Software Maintenance and Reengineering*, pages 133–143. IEEE, 2013.

[3] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. German, and P. Devanbu. The promises and perils of mining git. In *Mining Software Repositories, 2009. MSR'09*, pages 1–10. IEEE, 2009.

[4] C. Chilipirea, G. Laurentiu, M. Popescu, S. Radoveneanu, V. Cernov, and C. Dobre. A comparison of private cloud systems. In *Advanced Information Networking and Applications Workshops (WAINA)*, pages 139–143. IEEE, 2016.

[5] F. Frattini, R. Ghosh, M. Cinque, A. Rindos, and K. S. Trivedi. Analysis of bugs in Apache virtual computing lab. In *Int'l Conf. Dependable Systems and Networks (DSN)*, pages 1–6. IEEE, 2013.

[6] W. Jenkins, S. Vilkomir, P. Sharma, and G. Pirocanac. Framework for testing cloud platforms and infrastructures. In *Int'l Conf. Cloud and Service Computing (CSC)*, pages 134–140. IEEE, 2011.

[7] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101. ACM, 2014.

[8] R. Kumar, K. Jain, H. Maharwal, N. Jain, and A. Dadhich. Apache CloudStack: Open source infrastructure as a service cloud computing platform. *Int'l J. Advancement in Engineering technology, Management and Applied Science*, pages 111–116, 2014.

[9] M. M. Lehman. Programs, life cycles, and laws of software evolution. *Proc. IEEE*, 68(9):1060–1076, September 1980.

[10] A. K. Maji, K. Hao, S. Sultana, and S. Bagchi. Characterizing failures in mobile OSes: A case study with Android and Symbian. In *Int'l Symp. Software Reliability Engineering (ISSRE)*, pages 249–258. IEEE, 2010.

[11] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *Int'l Symp. Cluster Computing and the Grid*, pages 124–131. IEEE Computer Society, 2009.

[12] N. Palix, G. Thomas, S. Saha, C. Calvès, J. Lawall, and G. Muller. Faults in Linux: Ten years later. In *ACM SIGPLAN Notices*, volume 46, pages 305–318. ACM, 2011.

[13] S. Ramani, S. S. Gokhale, and K. S. Trivedi. Srept: software reliability estimation and prediction tool. *Performance evaluation*, 39(1):37–60, 2000.

[14] O. Sefraoui, M. Aissaoui, and M. Eleuldj. OpenStack: toward an open-source solution for cloud computing. *Int'l J. Computer Applications*, 55(3), 2012.

[15] A. Vogel, D. Griebler, C. A. Maron, C. Schepke, and L. G. Fernandes. Private IaaS clouds: a comparative analysis of OpenNebula, CloudStack and OpenStack. In *Int'l Conf. Parallel, Distributed, and Network-Based Processing (PDP)*, pages 672–679. IEEE, 2016.

[16] A. Wood. Predicting software reliability. *Computer*, 29(11):69–77, 1996.

[17] A. Zerouali and T. Mens. Analyzing the evolution of testing library usage in open source Java projects. In *Int'l Conf. Software Analysis, Evolution and Reengineering (SANER)*.