Green Computing and Software Defects in Open Source Software: An Empirical Study

Faheem Ahmed1, Hasan Mahmood2, Adeel Aslam3
Department of Computing Science1, Thompson Rivers University, Kamloops, Canada
Department of Electronics2,3, Quaid-e-Azam University Islamabad, Pakistan
fahmed@tru.ca1, hasan@qau.edu.pk2, aaslam@student.qau.edu.pk3

Abstract— Overtime open source software (OSS) is receiving attention from industry. The growth rate is unprecedented. The software industry is inclining towards adopting OSS despite concerns about the quality. Defect management has been a key issue in OSS. Software whether it is close or open source uses energy. Green computing argued that environmental conditions are getting worst and use of power consumptions should be minimized to protect the earth. A lot of work has been reported in the area of green hardware but very little work has been reported in making software green. The main objective this work is to increase understanding of green software development. In this study, we empirically investigate that there is relationship between defect management in OSS and power consumption of OSS application. We used a dataset consist of 82 OSS projects to study the research model of this investigation. The results of this investigation show that software defect management in OSS play a significant role in managing power consumption of OSS applications.

Keywords—Open source software, software engineering, software defect management, empirical software engineering, Green computing, Green software code

I. INTRODUCTION

Many organizations are considering OSS a viable option to close code software and making their code available as well participating in other existing OSS projects. The collaborative nature of OSS development environment has proven the reduction of maintenance efforts. OSS refers to the use of shared source code, open standards, and collaboration among software developers and users worldwide to build software, identify and correct errors, and make enhancements[1]. The essence of OSS development environment envisages a freely available code, which anyone can modify, and share the modified version with other people as well. The other aspects of OSS development are time to delivery, there is no timeline and code can be modified at any time. Software defect management in open source software deals with involvement of many independent users who have no direct interests in the software. These users test the software independently and identify the software defects in an early development phase, which ultimately reduces the cost of post development changes. The benefits of early software defect identification are that it reduces the post distribution challenges and issues as well any additional costs that may occur due to fixing the defects. Particularly in consumer, electronic industry, by using pre-made OSS components for low-level routine, will release resources to focus on higher-level components for competitive advantage[2]. Active open source projects usually have a well-defined community with common interests, which is involved either in continuously evolving its related products or in using its results[3]. Loosely organized communities of participants located around the world and working over the Internet develop OSS and remarkably, most participants contribute without being employed, paid, or recruited by the organization [4]. Internet further accelerates the concept of OSS development environment. The OSS community is renowned as a close interaction of professional and amateur software developers and the development character of OSS ensures that reuse is a central pillar in project development.

Computer programs are using considerable amount of energy during their execution cycle. Researchers are targeting to develop mechanism to minimize the use of energy in all facets of life. Green computing envisions using equipment and operations in a healthy and responsible manner in order to reduce the environmental footprint. In the recent past researchers, have more concentrated on the hardware aspects of computer, and proposed mechanism to make green hardware. Some examples are designing low power consuming microprocessors; minimizing heat generation, load management etc. On the other hand, software also contributes in energy consumption but has been given little attention.

A green and sustainable software product means that economic, societal, and ecological impacts, as well as impacts on human beings, that result from the product over its whole life cycle, should be as small as possible[5]. Juha[6] observes that from the sustainable development and climate change point of view, good software helps to reduce greenhouse gases, waste and resource requirements while bad software increases them. One of the existing research directions to reduce power consumption of software application is to develop green algorithms and data structures[7][8]. This paper aims to investigate the relationship between defect management in OSS development environment and power consumption of resulting OSS application.

a. Research Motivations & Related Work

Software maintenance has been a key phase of software life cycle because it deals with defect management. There are various types of maintenance activities such as perfective, corrective and enhancement. Vixie[9] finds that in OSS the software life cycle activities such as requirement definition,
system level and detailed design, unit and system testing, and support are not carried out in a manner similar to traditional software engineering. This also raises issues concerning defect identification and fixing which falls into the category of corrective maintenance. Much work has been reported in overall defect management in OSS.

Koponen[2] discuss defect management and version management system as an integral part of OSS maintenance process. Raymond suggests the high quality of OSS can be achieved due to high degree of peer review and user involvement in bug/defect detection. Mockus et al.[10] provides a comprehensive comparison of Apache against five commercial products in terms of developer participation, team size, productivity and defect density, and problem resolution. An open source software project in which many people are actively collaborating, exchanging ideas, identifying and giving suggestion to fix the bugs, requesting to expand the scope of the project is known as active or alive project.

A green and sustainable software product means that economic, societal, and ecological impacts, as well as impacts on human beings, that result from the product over its whole life cycle, should be as small as possible[11]. Naumann[5] green and sustainable software engineering is the art of defining and developing software products, so that the negative and positive impacts on sustainable development that result and/or are expected to result from the software product over its whole life cycle are continuously assessed, documented, and used for a further optimization of the software product. Capra et al.[12] observe that different software applications covering the same set of functional requirements differ significantly in their energy consumption. Tiwari et al. [13] conclude that ability to evaluate software in terms of power consumption makes it feasible to search for low power implementations of given programs; in addition, it can guide the development of general tools and techniques for low power software. Hindle[14] proposes a methodology of relating software change to software power consumption and we applied this methodology in two case studies on two distinct systems. Ellis[15] advocates that reducing energy consumption should be raised to first-class status among performance goals when software is being designed.

No work has been reported in the areas of relating software defect management and power consumption of OSS application. This is the first study at the best of our knowledge to address this significantly important aspect of OSS and green code development.

II. RESEARCH MODEL AND HYPOTHESIS OF THE STUDY

OSS projects are gaining popularity and growth rate is exceptionally high, which shows the trust of software development community in the development environment. The use of internet further has created virtual community, which are geographically apart but still working together like a team. The main mode of communication is the online forums where the community shares the knowledge. Software defects are also first pointed out in the online forums and then possible solutions start coming up as the discussion prolong. It shows that community is interested in gaining benefits of a software when people are constantly exchanging messages in the online forum to help identifying and fixing software defect. The main objective of the research model shown in Figure 1 is to analyze the association between OSS software defects identification and fixing and power consumption of the resulting OSS application. The main objective of this study is to investigate the answer to the following research question:

RQ: Does software defect management helps in lowering the power consumption of OSS application.

In order, to empirically investigate, the research question we hypothesize the following:

H-0: The number of bugs fixed in OSS project is positively related with power consumption of OSS application.

It is important to mention here that we are using the term “fixed bug” as a defect, which is identified, reported and fixed.

a. Data Collection and Experimental Setup

We collected the data of 250 open source software projects from www.sourceforge.net, a popular data repository of open source software projects on the internet. The dataset covers the category of “business and enterprise”. The first filtration activity removes the data of all those projects, which has fixed bugs of 0. This reduces the dataset to 82 projects. Figure 2 illustrates the number of bugs fixed in various open source software project of the final dataset of this study. Figure 3 illustrates the programming language distribution of the dataset. Majority of the projects are developed in either Java or combination of languages. Figure 4 illustrates the traffic distribution of study dataset. It illustrates that majority of the projects have reasonable traffic.

A traffic count illustrates how many site visits has been reported. In general low traffic OSS projects tend to die off whereas other has the tendency of bug fixing, support and new feature addition. Figure 5 highlights the userratings. It illustrates that majority of the projects are highly rated by the users. The language distribution includes C (8), C# (6), C++ (6), VB (2), Java (25), Multi (27), Per (1), PHP (5), Phyton (2) projects. A total of 12 projects have the highest rating of 5, whereas 56 projects have the ratings between 4 and 5. Only one project has rating below 3.

We collect quantitative data about the power consumption of the OSS application. We used the Joule meter [16][17] to collect data about the power consumption of the OSS application. The Joule-meter estimates the power usage through a power model that relates the computer resource usage and hardware power state (processor utilization, processor frequency, screen brightness, monitor on/off state, disk...
utilization) to power drawn, however, the power usage of applications shows only the power consumed by the application on the CPU[17]. We downloaded freely available OSS application from the sourceforge.net to include in this study. We assigned an individual to operate all these OSS applications. The individual operated has no experience in operating these applications before. The individual initially spent some time to learn the application. Since different execution scenarios (features of a program) of a program may result in different power consumption profiles, therefore we recorded the average power consumption of the application after collecting several measurements at various intervals of time. We tried to execute all the possible features of the program. We recorded the average application power consumption after collecting several measurements at various intervals of time. The specific intervals of recording power use are handled by the Joule meter application.

To analyze the research model and check the significance of hypothesis H-0 we used various statistical analysis techniques. Initially we divided the data analysis activity into three phases. Phase-I dealt with normal distribution tests and parametric statistics. Phase-II dealt with non-parametric statistics. In order to increase the external validity of the study, we used both statistical approaches of parametric and non-parametric methods. We tested for the normal distribution of fixed bugs using mean, standard deviation, kurtosis and skewness techniques, and found the values for all these tests to be within the acceptable range for the normal distribution with some exceptions.

We conducted tests for hypotheses H-0 using parametric statistics, such as the Pearson correlation coefficient and one tailed t-test in Phase-I. In Phase-II of non-parametric statistics, we conducted tests for hypothesis using the Spearman correlation coefficient. Phase-III dealt with testing the hypothesis of the research model of this study using the technique of Partial Least Square (PLS). The PLS technique helps when complexity, non-normal distribution, low theoretical information, and small sample size are issues. In the PLS testing of hypothesis, we keep fixed bugs as independent and amount of power consumption as dependent variable. We used the PLS technique to increase the reliability of the results. The statistical calculations were performed using Minitab® 14 software.
PLS technique to overcome some of the associated limitations. We placed one variable as the response variable and other variables as the predictor. We observed path coefficient, $R^2$, and F-ratio. The path coefficient of fixed bugs (H-0) was found to be $-0.017$, $R^2$: $0.05$ and F-ratio ($4.31$) was significant at $P < 0.04$.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Pearson Coefficient (p-value)</th>
<th>Spearman Coefficient (p-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H0</td>
<td>$-0.226$ ($P &lt; 0.04$)</td>
<td>$-0.160$ ($P &lt; 0.05$)</td>
</tr>
<tr>
<td>Path Coefficient</td>
<td></td>
<td>F-Ratio</td>
</tr>
<tr>
<td>-0.017</td>
<td>0.05</td>
<td>4.31</td>
</tr>
</tbody>
</table>

### III. DATA ANALYSIS & RESULTS

We examined the Pearson correlation coefficient and t-test between variables involved in the hypothesis H-0. Table 1 illustrates the results of various structure analysis tests we conducted to test the hypothesis. The Pearson correlation coefficient between fixed bugs and amount of power consumption of OSS application was negative ($-0.226$) at $P < 0.04$, and thus provided a justification to reject the H-0 hypothesis. In Phase-II we conducted non-parametric statistical technique using Spearman correlation coefficient to test the hypothesis H-0. A negative association ($-0.160$) at $P < 0.05$ was observed between fixed bugs and amount of power consumption of OSS application (H-0). Therefore, we reject the hypothesis H-0. In Phase-III of hypothesis testing, we used the PLS technique to overcome some of the associated limitations and to cross validate with the results observed using the approaches of Phase-Phase-II and I. We tested the hypothesized relationship, i.e. H-0 by examining its direction and significance. The hypothesis involves two variables therefore in PLS we placed one variable as the response variable and other as the predictor. We observed path coefficient, $R^2$ and F-ratio. The path coefficient of fixed bugs (H-0) was found to be $-0.017$, $R^2$: $0.05$ and F-ratio ($4.31$) was significant at $P < 0.04$.

### IV. DISCUSSION OF EMPIRICAL EVIDENCE

The collaborative nature of the OSS development environment envisages reducing the software defects and providing effective change management by opening up the communication channels. The development environment comprises of user forums, mailing lists, exchange of messages to fix the bugs or adding more functionalities. It is clear from our analysis that there is a negative correlation between the number of bugs fixed and amount of power OSS application consumes. This demonstrates that the OSS community is active in testing of projects and the identification of defects which further facilitates the power consumption of applications. Since power consumption of an application is related with the defect management activity, therefore in such cases defect management is not a simple activity that helps only in identifying and fixing bugs but also assists in developing a green software product.

Software defect identification and tracking is one of the main characteristics of the OSS development environment. Participants in the OSS project work collaboratively to identify and correct defects thus allows a comprehensive testing of software. A healthy discussion about understanding the software defect by studying the operating environment, pre and post conditions, program structure, programming language constraints and hardware resources etc. provides feasible solutions. The extensive review of the free available source code by volunteers provides a better opportunity of defect identification. Many of the contributing developers in OSS project community work in different OSS projects simultaneously or has worked in past therefore the potential of reusing code is relative high. RQ deals with finding the impact of software defect management on the power consumption of application. The investigation yields that the number of fixed software defects is negatively related with the power consumption. Defect removal is not just a coding error that is fixed, sometimes it is logical error which have an impact on the design of the software. If software has many defects that reflect, it has poor design and thus causing more power to consume.

### V. CONCLUSION

Free and open source software is gaining popularity due to many reasons at an unprecedented rate of growth. Organizations despite some concerns about the quality have been using them. The objective of this study was to analyze empirically the association between software defects and amount of power an application is consuming. We observed that number of fixed software defect is negatively related with the amount of power OSS application is consuming. The whole activity of software defect identification, feasible solution, and implementation of solution scenarios are describe and discuss in online forums which is an integral part of OSS development environment. This study further helps in understanding the development environment of OSS and its contributions in making software green.

### REFERENCES


