Defect Management Practices and Problems in Free/Open Source Software Projects

Dr. Anu Gupta anugupta@pu.ac.in
Assistant Professor
Department of Computer Science and Applications
Panjab University, Chandigarh, 160014, India.

Dr. R.K. Singla rksingla@pu.ac.in
Professor
Department of Computer Science and Applications
Panjab University,Chandigarh, 160014, India.

Abstract

With the advent of Free/Open Source Software (F/OSS) paradigm, a large number of projects are evolving that make use of F/OSS infrastructure and development practices. Defect Management System is an important component in F/OSS infrastructure which maintains defect records as well as tracks their status. The defect data comprising more than 60,000 defect reports from 20 F/OSS Projects is analyzed from various perspectives, with special focus on evaluating the efficiency and effectiveness in resolving defects and determining responsiveness towards users. Major problems and inefficiencies encountered in Defect Management among F/OSS Projects have been identified. A process is proposed to distribute roles and responsibilities among F/OSS participants which can help F/OSS Projects to improve the effectiveness and efficiency of Defect Management and hence assure better quality of F/OSS Projects.

Keywords: Free Software, Open Source, Defect Management, Quality, Metrics

1. INTRODUCTION

Free/Open Source Software (F/OSS) is an evolving paradigm of software development which allows the entire Internet community to use its combined programming knowledge, creativity and expertise to develop software solutions, which could render benefits to whole community without involving cost and proprietary issues [1]. F/OSS participants rely on extensive peer collaboration through the Internet using Version Control System, Mailing List, Defect Management System, Internet Relay Chat, Discussion Forum etc. [2]. These tools enable participants to collaborate in the F/OSS development process as well as act as repositories to store the communication activities of the participants, manage the progress and evolution of F/OSS Projects. These repositories contain explicit and implicit knowledgebase about F/OSS projects that can be mined to help developers in improving the product as well as to facilitate the users in evaluating the product.

Even though there are number of qualitative and quantitative studies about F/OSS, little attention has been paid to the rich information stored in Defect Management Systems of F/OSS projects [3-8]. Defect Management System provides an effective mechanism for recording and tracking of defects as well as promotes user involvement and peer review process. All the users may not have knowledge to participate in the development or code review of an F/OSS Project but such users may report bugs or request new features. They may also comment on existing defect reports or help in their removal, for example by reproducing them or supplying more information. A large amount of defect related data flows back and forth between the developers and the users of the F/OSS projects. Hence in most of the F/OSS projects, substantial amount of defect data gets accumulated in the Defect Management Systems over the period. This valuable defect data can be used to analyze the past experience, degree of improvement or deterioration in resolving defects and determine responsiveness towards users. As the potential F/OSS users need to evaluate the extensibility and maintainability before taking any decision to adopt a particular F/OSS product, so the defect related analysis can greatly help them to evaluate how efficiently and effectively the requests for fixing bugs, enhancing features, translation requests, support requests etc. are being managed. Moreover the availability of huge amount of information with a great variety in size, programming languages, tools, methods etc. offers the possibility of creating a comparison framework among F/OSS projects from which knowledge and experience can be gained.

In the current study, the defect data of various F/OSS projects is analyzed from various perspectives, with special focus on evaluating the efficiency and effectiveness in resolving defects and determining responsiveness towards users. Based on the findings, effective ways and means are suggested to improve defect management and thus enhance the quality of F/OSS projects.

2. F/OSS PROJECT SELECTION AND DATA COLLECTION

F/OSS projects are selected from SourceForge, a centralized place for F/OSS developers to host their projects [9]. It is the world's largest F/OSS projects repository with more than 230,000 F/OSS projects and over 2 million registered users. It provides some of the best empirical data on F/OSS research. A single source is chosen to select projects in order to control for differences in available tools and project visibility. In spite of large number of projects hosted, only a small proportion of these projects are actually active. Also many of the F/OSS projects either do not use or do not allow public access to Defect Management System. Hence those projects are considered for which defect related data is publicly accessible and is being maintained completely at SourceForge. Another criterion used for selection of projects is the project development stage (1-6 where 1 is the planning and 6 is a mature stage). A cut-off of 5 is chosen which indicates that the selected projects are at similar stage of development and are not in the early stage of development lifecycle. A total of 20 projects are selected which constitute a diverse mix of project size, team size, nature of application and targeted end user type. Selection of limited number of projects has helped to carry out in-depth study. For all the selected F/OSS projects, detailed defect data is downloaded from SourceForge Research Data Archive (SRDA) for the period starting from their respective Registration Date to October 2008 [10]. The defect data is downloaded on the basis of unique Project ID assigned to each project at SourceForge and is stored in the local repository (mySQL) aggregating more than 60,000 defect records. Further the Defect Analysis and Reporting Tool (DART) is used to carry out exhaustive analysis of defect data and generate variety of textual/graphical reports. For selected F/OSS projects, various parameters used for analyzing defect data and their quantitative results are discussed in subsequent sections.

3. Quality Metrics used for evaluating Defect Management

In order to evaluate Defect Management among F/OSS projects, various metrics used are mentioned in Table 1.

Sr. #	Metric Name	Formula	Objective
1.	Defect Resolution	Cumulative Defect Arrival Pattern and Defect Closure pattern over time interval (in months)	To check consistency and efficiency in defect resolution over the period
2.	Pending Defects	Frequency as well volume of increase/decrease in pending defects over period (in months)	To check the trend of pending defects over the period
3.	Defect Removal Rate	Proportion of defects resolved out of defects submitted for a particular period	To observe the rate at which defects are resolved over the period
4.	Backlog Management	Ratio of number of defects closed to number of defects arrived during the period	To measure the capability to handle the pending defects
5.	Software Release and Backlog Management	Tracing the shapes of BMI curves with release history of the project	To observe the relationship of software releases with defect handling over the period
6.	Defect Resolution Age	Number of days elapsed since a defect arrived till the time defect is resolved/closed.	To measure the resolution efficiency
7.	Fix/Non-Fix Defect Resolution	Defect Resolution Age for Fix versus Non-Fix Defects	To compare the efficiency in handling defects requiring code change with defects requiring no code change
8.	Defect Pending Age	Number of days elapsed since a defect arrived and still remained pending at the end of the month	To measure the age of pending defects at any point of time
9.	Defect Resolution (Defect Type Wise)	Defect Resolution Age for Bugs versus Feature Requests versus others	To compare the efficiency in handling defects belonging to various defect types

TABLE 1: Quality Metrics used for Evaluating Defect Management

4. QUANTITATIVE RESULTS

The detailed results obtained are being presented with the help of statistics and various graphs in the following subsections.

4.1. Defect Resolution

Defect arrival curves and defect closure curves have been drawn for all the selected F/OSS projects on the basis of live defect data consolidated on monthly basis. Defect arrival curve is related to the defects reported by F/OSS community, represented as Cumulative Defects arrived over the period. Defect closure curve is related to the resolution and closing of defects by F/OSS community, represented by Cumulative Defects closed over the period. The distance between these two curves at a given point in time represents the number of defects pending at that time. An ideal defect resolution process needs to be

- **Continuous:** when cumulative closed curve is quite smooth without having any peaks or steps.
- **Efficient:** when cumulative closed curve stays near to the cumulative open curve without raising overall number of pending defects.

The graphs for all the selected F/OSS projects have been drawn which show varying patterns. Those patterns can be classified among the following four categories [11]:

- Continuous and Efficient
- Discontinuous and Efficient
- Continuous and Inefficient
- Discontinuous and Inefficient

The patterns for all the selected F/OSS projects are identifiable in one or the other category and helpful in determining the quality of defect resolution process. The example graphs for each of the above categories are shown in Figure 1 to 4.

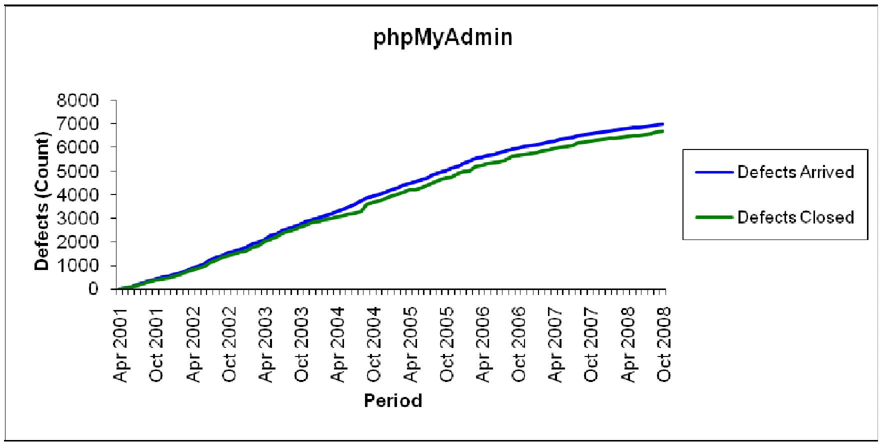

FIGURE 1: Continuous and Efficient Defect Resolution

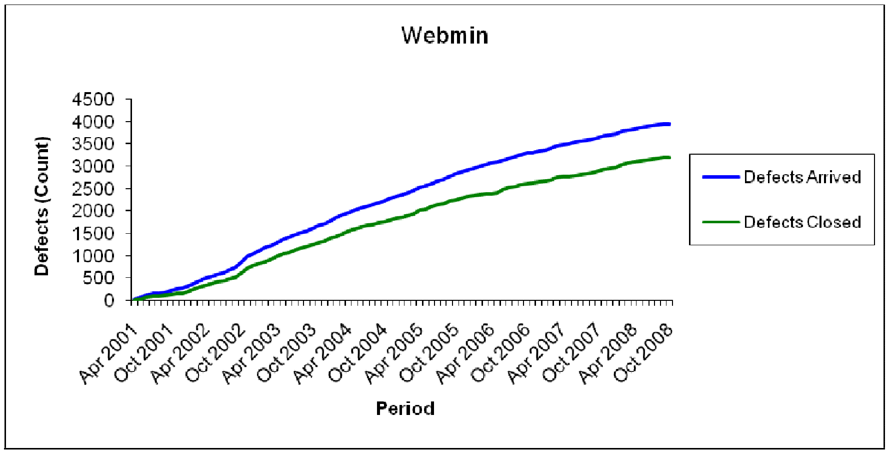

FIGURE 2: Continuous and Inefficient Defect Resolution

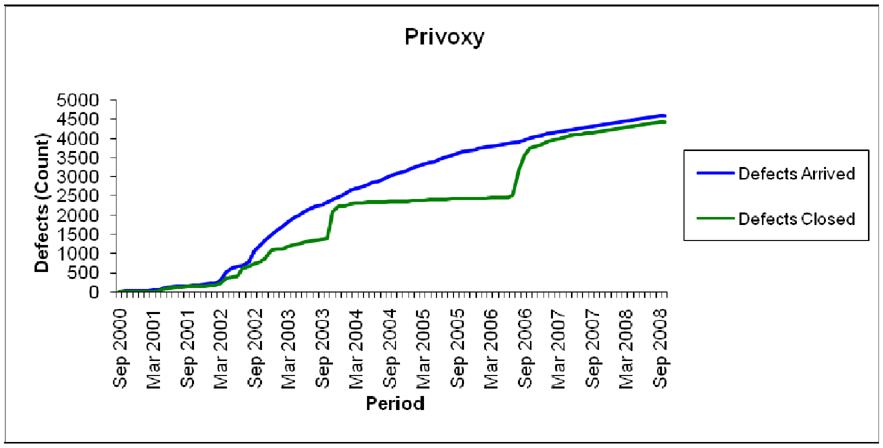

FIGURE 3: Discontinuous and Efficient Defect Resolution

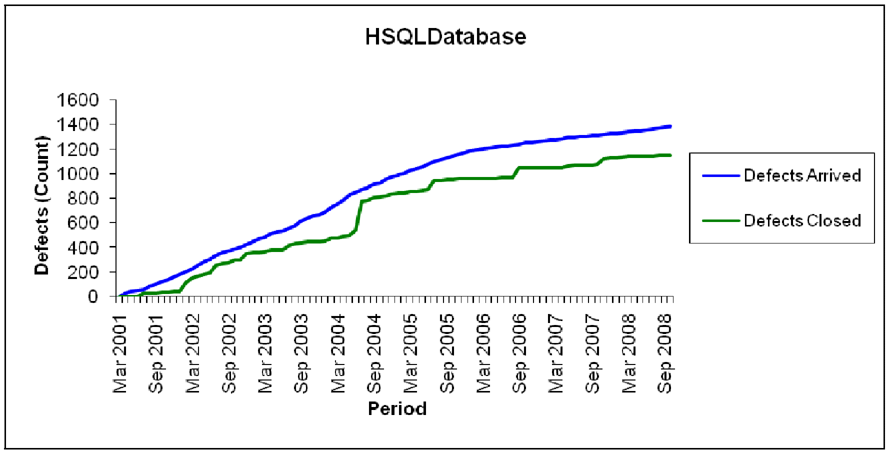

FIGURE 4: Discontinuous and Inefficient Defect Resolution

4.2. Pending Defects

Pending defects refers to all those defects which still need to be addressed. Ideally pending defects should decrease with the passage of time or at least it should remain constant. Large number of pending defects may discourage participating users from providing further feedback and many opportunities of improvement in the software may be lost. Figure 5 shows that number of monthly pending defects for all the 20 projects taken together keeps on increasing. To confirm the same statistically, a paired two-sided *t*-test is applied between number of pending defects in the beginning and at the end of the observation period for each of the 20 projects. It is clearly seen that there is significant difference (*t*(19)=3.93634888, p<0.05; *t* critical =2.09302405).

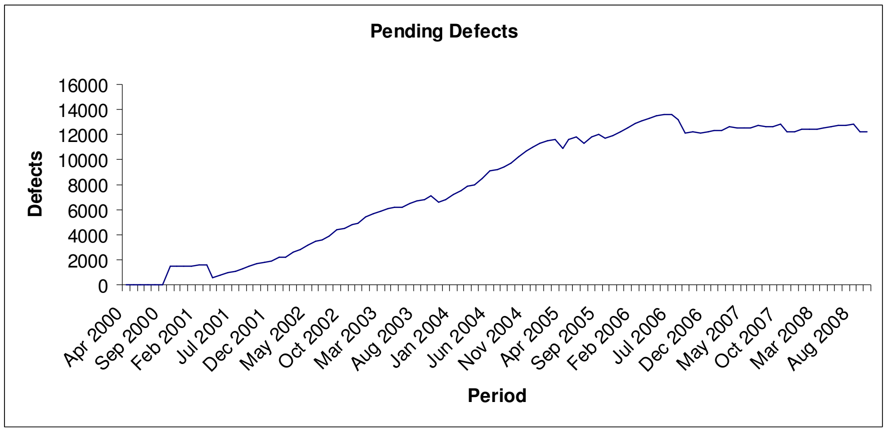

FIGURE 5: Aggregate Pending Defects for 20 F/OSS projects Together

The closer examination of pending defects over the period January 2006- November 2007 (Figure 6) shows that there are usually gradual increases and steep decreases in the number of pending defects. This suggests that defects slowly accumulate over the period and are removed in bursty manner. To test the hypothesis statistically, changes in pending defects from one month to the next month are recorded in form of upward change (for an increase) and downward change (for a decrease) frequencies for each of the 20 projects. Paired two-sided *t*-test shows that the difference between upward and downward changes in the number of pending defects is significant (*t*(19)=11.9702; p<0.05; *t* critical = 1.7291). There are overall 2.91 times more upward changes than downward changes. On an average basis, whenever there is an increase in pending defects, the upward change is 16.63 defects per month. On the other hand, if pending defects decrease, the downward change is 30.36 defects per month on an average. The reason for bursty nature of defect resolution is further discussed in subsection 4.5.

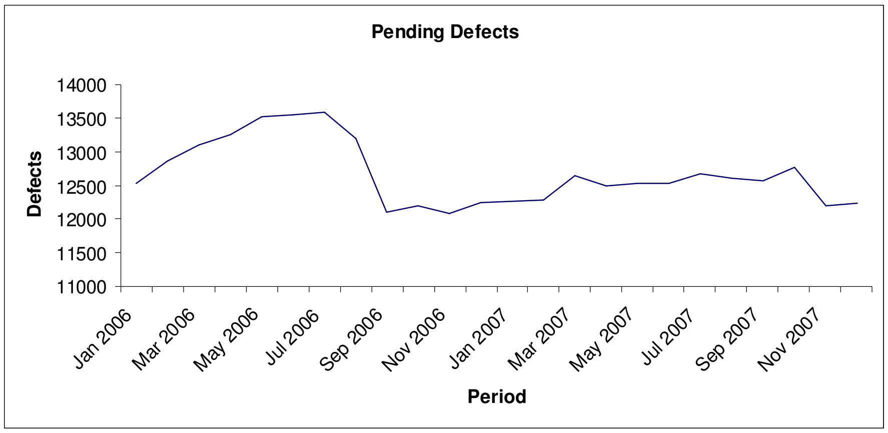

FIGURE 6: Gradual Increases and Steep Decreases in Pending Defects

4.3. Defect Removal Rate

Defect removal rate refers the proportion of defects resolved out of defects submitted for a particular period. The ever increasing number of pending defects indicates that the defect removal rate is decreasing. The size of core team has remained roughly same among the selected F/OSS projects. The hypothesis is that certain percentage of defects does not get resolved over the period as defect reports are submitted, thus number of pending defects accumulate.

In order to investigate this hypothesis statistically, a period of five years from 2003 to 2007 is considered. For each selected project, all the defects reports submitted during a particular year have been considered and then the status of each defect report exactly after 1 year of defect submission is observed whether the defect is resolved/closed or not [12]. The application of ANOVA reveals that the period in which a defect is submitted has significant influence on the defect removal rate ($F(4,94)=6.058928$; $p<0.05$; F critical=2.468533). The defects that have been reported during the year 2003, 81% of them have been resolved after 1 year (Table 2). The defect removal rate reduces to 71% in year 2005 and further to 65% in year 2007. This clearly shows that the defect removal rate is declining which results in ever increasing number of pending defects.

Period	Average Removal Rate	Standard Deviation
Year 2003	0.81	0.11
Year 2004	0.74	0.20
Year 2005	0.71	0.15
Year 2006	0.66	0.23
Year 2007	0.65	0.26

TABLE 2: Defect Removal Rate Over Five Years

4.4. Backlog Management

Backlog management refers to the capability of F/OSS developers to handle the pending defects, measured using Backlog Management Index (BMI). BMI is a ratio of number of defects closed to number of defects arrived during the period.

$$BMI = \frac{Number of\ defects closed\ during the\ period}{Number of\ defects arrived\ during the\ period} \times 100$$

If BMI is larger than 100, it means that the backlog is reduced as defects are being closed at the same or higher rate at which the defects are arriving. If BMI is less than 100, the backlog is increased. Of course, the goal is always to strive for a BMI larger than 100. With enough data points, the technique of control charting can help to calculate the overall backlog management capability of the software process [13]. A control chart is a graph or chart with limit lines, called control lines. In fact BMI chart is a pseudo-control chart because BMI data are auto correlated and assumption of independence for control charts is violated. As the BMI values are in wide range, c control chart is more suitable [13]. In this case, three kinds of control lines are calculated as follows:

- Central Line (CL) equal to Mean BMI
- Lower Control Limit $LCL = (CL - 3 \times \sqrt{\mathrm{CL}})$
- Upper Control Limit $UCL = (CL + 3 \times \sqrt{\mathrm{CL}})$

If a process is mature and under statistical process control, all values should lie within the LCL and UCL. If any value falls out of the control limits, the process is said to be out of statistical process control. Figure 7 shows a project having very good backlog management. Most of the times the BMI curves are able to maintain themselves above the LCL. In case of Figure 8, the project was not having good backlog management initially but later on it improved. Figure 9 shows poor backlog management throughout the period.

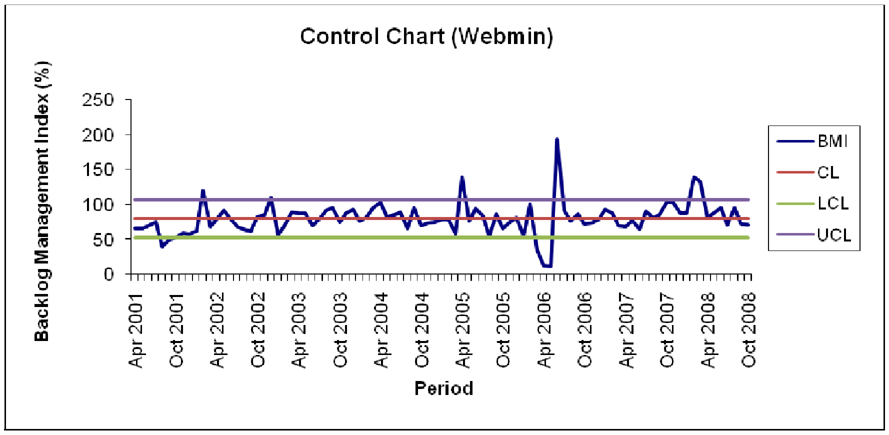

FIGURE 7: Backlog Management of Defects (Good)

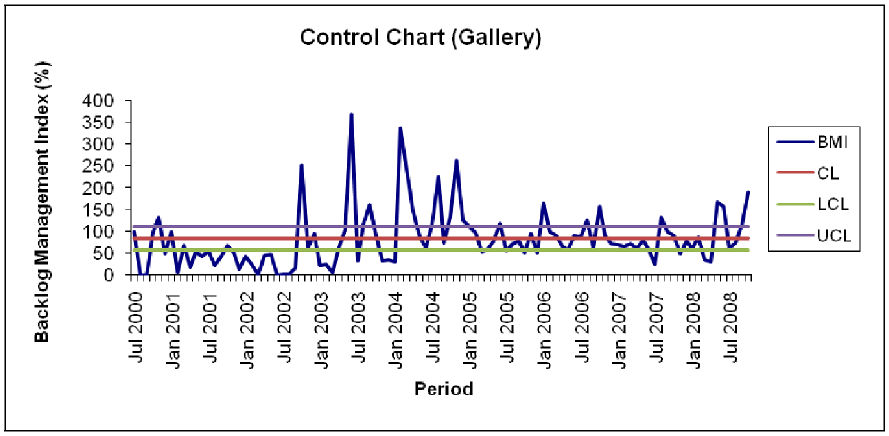

FIGURE 8: Backlog Management of Defects (Improved Later)

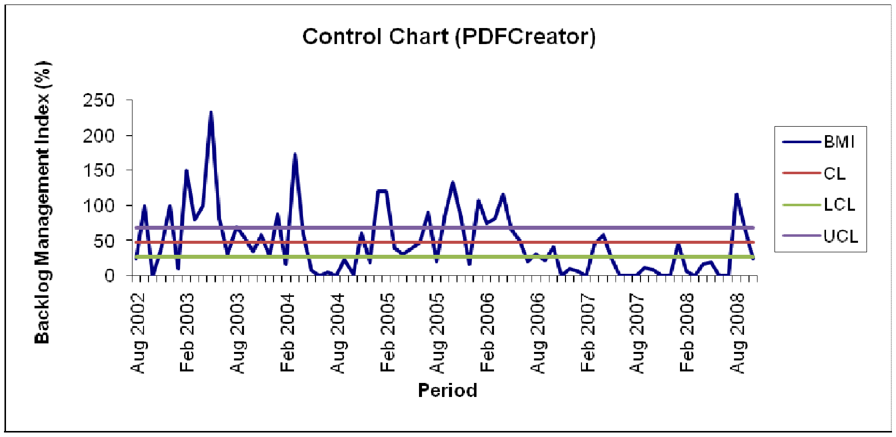

FIGURE 9: Backlog Management of Defects (Poor)

4.5. Software Release and Backlog Management

In the subsection 4.4, it is observed that BMI curves for most of the F/OSS projects are very fluctuating in spite of the fact that BMI values remain greater than 100 or lesser. To find out the reasons for such behavior, a detailed analysis of release data with BMI curves was carried out. Detailed inspection of release data revealed that the F/OSS projects are releasing their minor/major versions very frequently confirming the premise "*Release Early, Release Often*" [1]. In the Figure 10 and 11, efforts are made to trace back the shapes of BMI curves with release history of the projects. The dotted red colored vertical lines are drawn corresponding to major/minor releases in each of the following graphs.

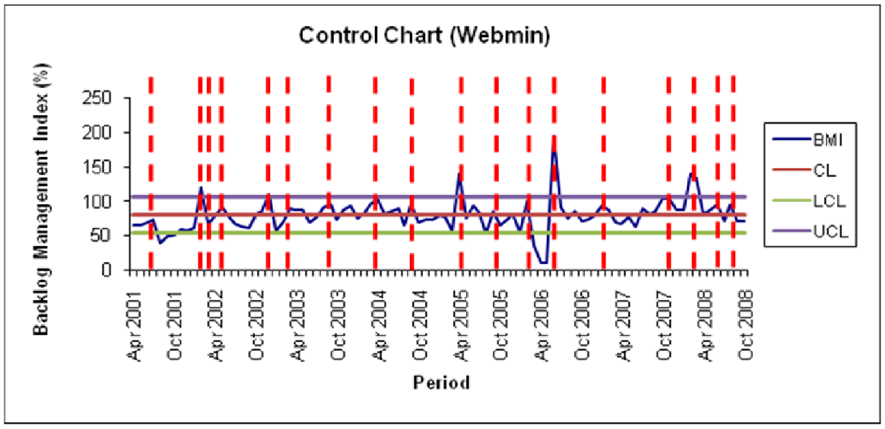

FIGURE 10: Software Release and Backlog Management of Defects

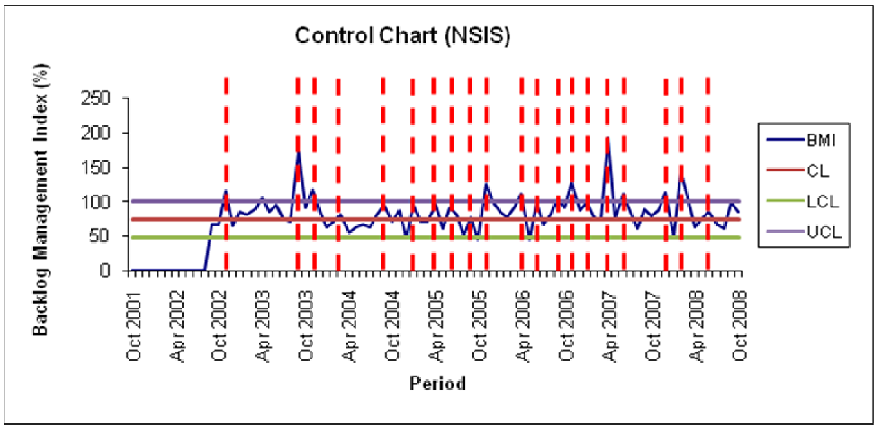

FIGURE 11: Software Release and Backlog Management of Defects

It is found that more than 90% of spikes in BMI curves are matching with the version releases. This phenomenon refers that generally F/OSS developer community do not resolve the defects on regular basis, instead put additional efforts to resolve defects near to each release.

4.6. Defect Resolution Age

Defect Resolution Age (DRA) refers to the number of days elapsed since a defect arrived till the time defect is resolved/closed. The average defect resolution age should be short as well as quite consistent to have efficiency in defect resolution. The monthly average of defect resolution age (MADRA) is computed using the following formula:

$$DRA(d_i)=\text{Defect Closing Date}(d_i)-\text{Defect Opening Date}(d_i)$$

$$MADRA = \frac{\sum_{i=1}^{N} DRA(di)}{N}$$ Where d_i refers to a defect closed

The graphs are drawn to show curves for average defect resolution age over the period for the F/OSS projects. Corresponding linear trend lines are also plotted. The projects should have preferably decreasing or at least constant trend of average defect resolution age to bring efficiency in defect resolution. For the F/OSS projects under study, it is observed that none of the projects has decreasing trend, very few projects are having near to constant trend lines (Figure 12) and most of the projects are showing upward trends in average defect resolution age over the period (Figure 13).

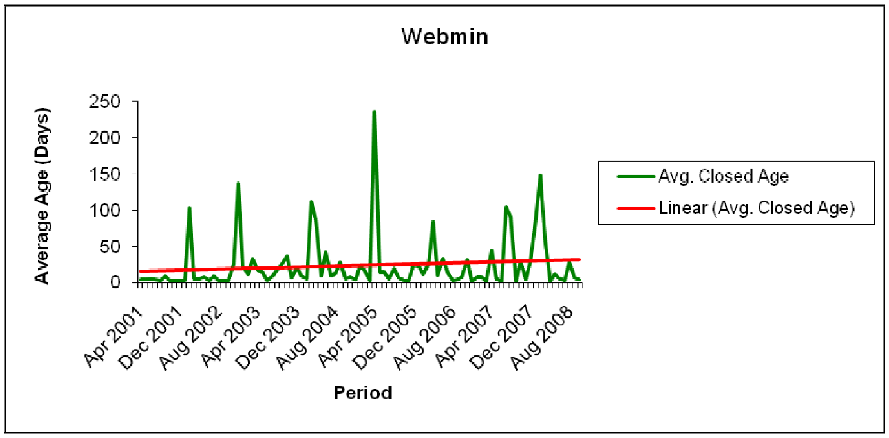

FIGURE 12: Defect Resolution Age (Near to Constant Trend)

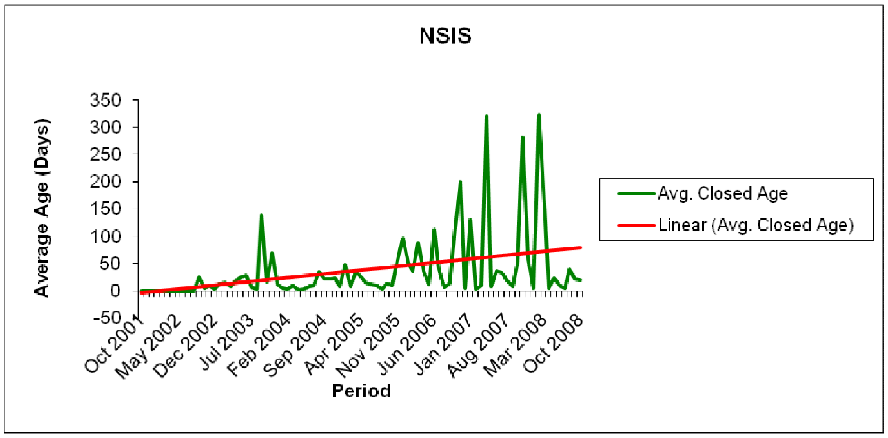

FIGURE 13: Defect Resolution Age (Increasing Trend)

Project	Period	Average Resolution Age (Days)	Standard Deviation	ANOVA Statistics
Webmin	Jan.1, 2002 to Dec. 31, 2003	21.56	32.46	*F*(2,69)= 0.220411; p<0.05; *F* critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	28.90	51.62	
	Jan.1, 2006 to Dec. 31, 2007	23.89	29.62	
NSIS	Jan.1, 2002 to Dec. 31, 2003	16.48	30.25	*F*(2,69)=7.176098; p<0.05; *F* critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	19.98	21.48	
	Jan.1, 2006 to Dec. 31, 2007	69.61	86.51	

TABLE 3: One Way ANOVA Statistics on Defect Resolution Age

To confirm the observation about trends in Defect Resolution Age, a standard analysis of variance (ANOVA) is carried out on monthly average resolution age over the period for all the selected F/OSS projects. Statistics about two projects are shown in Table 3. It is clearly seen that there is no significant difference in the average resolution over the period in case of Webmin, while it differs significantly for NSIS.

To analyze the overall defect resolution age for all the selected projects together during the investigation period, average resolution age for each of the 20 projects for various years is taken into consideration and standard analysis of variance (ANOVA) is applied which shows that there is significant change in defect resolution age over the period (*F* (4,94) =4.29461975;p<0.05;*F* critical =2.468533). The Table 4 also shows a continuous increasing trend in average defect resolution age (days) for various years for all the 20 projects taken together.

Period	Average Defect Resolution Age (Days)	Standard Deviation
2004	61.77	58.34
2005	76.35	41.62
2006	98.73	71.95
2007	113.07	89.99
2008	149.53	104.62

TABLE 4: Average Defect Resolution Age for 20 F/OSS projects Together

Figure 14 is a scatter plot for one of the F/OSS projects where each point represents resolution age for each defect closed. While Figure 15 shows the number of defects resolved with same resolution age value. The quality of the defect resolution process can be quantified by considering two statistical indices of the resolution age distribution i.e. skewness and kurtosis [51]. Skewness measures the asymmetry of the distribution and high values indicate that there are certain defects which have resolution age much higher than the average one. While Kurtosis measures the peaked ness of the distribution and high values mean that the variance of the resolution age is caused by very few defects with extremely long closing time (Table 5).

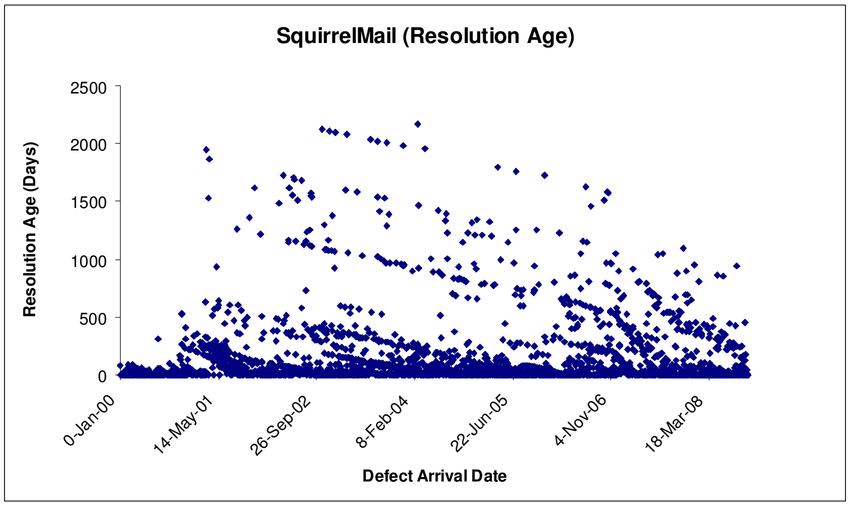

FIGURE 14: Scatter Plot of Resolution Age

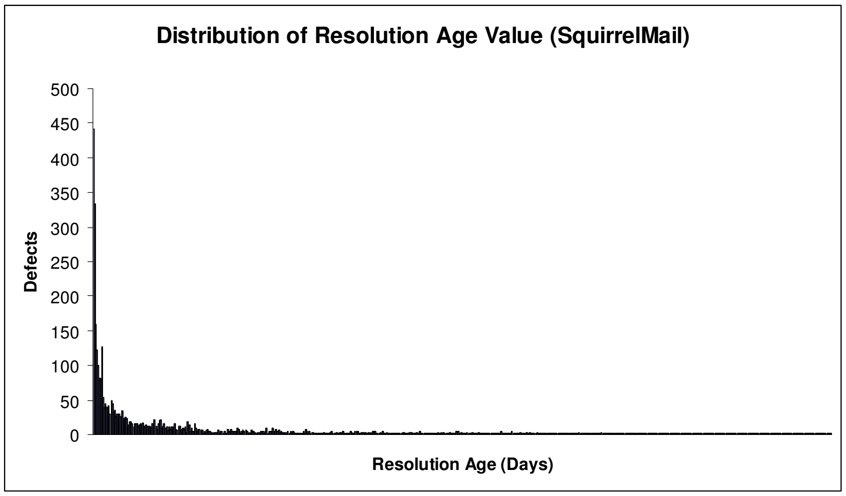

FIGURE 15: Distribution of Resolution Age

	SquirrelMail	NSIS	Webmin
Mean	6.20	7.02	13.09
Standard Deviation	25.29	37.83	100.33
Kurtosis	188.75	131.18	127.30
Skewness	12.58	10.97	11.04
Sum of Resolved Defects	3880	1362	3194

TABLE 5: Descriptive Statistics on Distribution of Resolution Age

It is clearly indicated that in most of the selected F/OSS projects, larger number of defects are resolved in shorter period while smaller number of defects are resolved in longer period which leads to an increase in overall mean resolution age.

4.7. Fix/Non-Fix Defect Resolution

It is observed that there is generally an increasing trend in defect resolution age and some of the defects are even resolved after 365 days. Many defects are resolved by making change/fix in the source code whereas others may be resolved with non-fix status such as Invalid, Won't fix, Out of date, Duplicate, Works for me, Rejected etc.

Hence further analysis is carried out by comparing the resolution age in fix and non-fix categories. Figure 16 and 17 show graphs for two of the selected projects where comparison is made between fix and non-fix resolutions by distributing the resolved defects on the basis of defect resolution age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days). It is found that even the defects with non-fix resolution are closed in higher ranges of resolution age i.e. 91 to 365 days or More than 365 days. It is also observed that the proportion of non-fix resolved defects remain more or less same across all the resolution age categories.

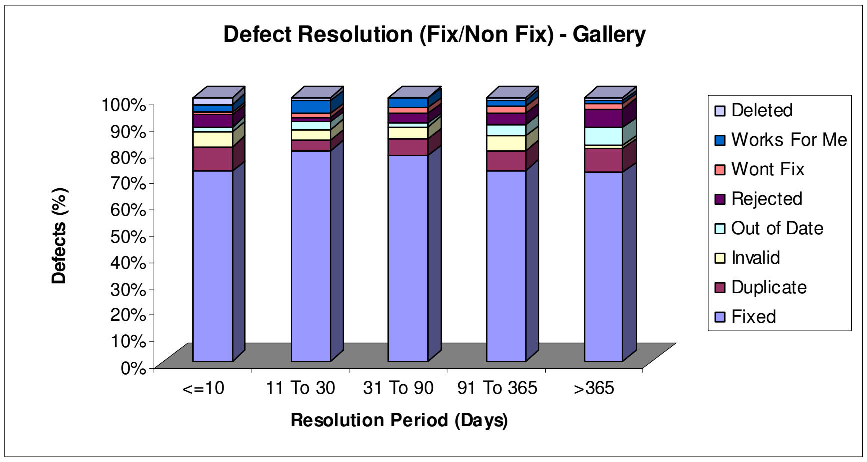

FIGURE 16: Defect Resolution Fix/Non-Fix (Gallery)

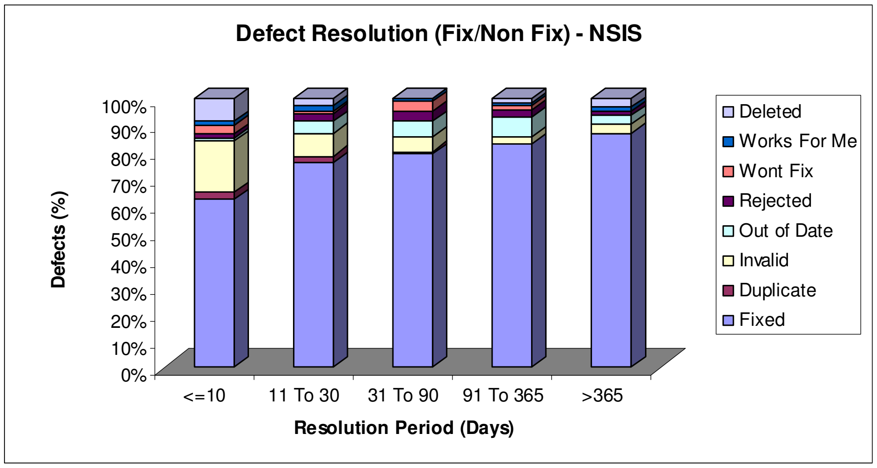

FIGURE 17: Defect Resolution Fix/Non-Fix (NSIS)

An unpaired two-sided *t*-test is conducted between defects with fix and non-fix resolution using their monthly average resolution age over all the months. The *t*-values in the last column of Table 6 for various F/OSS projects are below the critical values which clearly show that there is no significant difference in the resolution age of fix and non-fix resolved defects. An unpaired two-sided *t*-test is also applied to overall average age of defects with fix and non-fix resolution for all the 20 F/OSS projects. The test statistics ($t(38)=0.984940769$; $p<0.05$; t Critical=1.685954461) shows that there is no difference in efficiency for defects with fix and non-fix resolution as a whole also.

Project	Resolution Type	Average Resolution Age(Days)	Standard Deviation	*t* value*
Squirrelmail	Fix	132.58	165.75	0.820547
	Non-Fix	106.78	260.96	
Gallery	Fix	104.66	110.02	0.65281
	Non-Fix	117.43	167.31	
Webmin	Fix	22.47	37.78	0.21268
	Non-Fix	24.05	75.40	
NSIS	Fix	43.12	85.56	2.123759
	Non-Fix	21.41	47.08	
Netwide Assembler	Fix	132.13	308.57	0.645107
	Non-Fix	101.68	278.20	
aMSN	Fix	62.91	69.01	0.572375
	Non-Fix	55.69	81.39	

*p<0.05

TABLE 6: *t*-test statistics on Defect Resolution (Fix/Non-Fix)

4.8. Defect Pending Age

Defect Pending Age (DPA) refers to the number of days elapsed since a defect arrived and still remained pending at the end of the month. For all the selected F/OSS projects, monthly average of defect pending age (MADPA) is computed using the following formula:

$$DPA(d_i)=Current\ Date-Defect\ Opening\ Date(d_i)$$

$$MADPA = \frac{\sum_{i=1}^{N} DPA\ (di)}{N}$$ Where d_i refers to a pending defect

The graphs are plotted to show the curves for monthly averages of defect pending age for all the projects. It is observed that all the projects are showing increasing trend of monthly average defect pending age. Further detailed analysis of defects pending age is carried out by distributing the pending defects according to their pending age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days). Figure 18 and 19 also show curves for the overall monthly average pending age of all the pending defects as well as monthly average pending age for defects falling in each of the categories. By observing the pattern of defect pending age over the period, it is found that in almost all the projects the average pending age is increasing. But this increase in defect pending age trend is attributed mainly by those defects whose average pending age is 90 days or more. While in other lower age categories, trend remains either constant or slightly downward/upward.

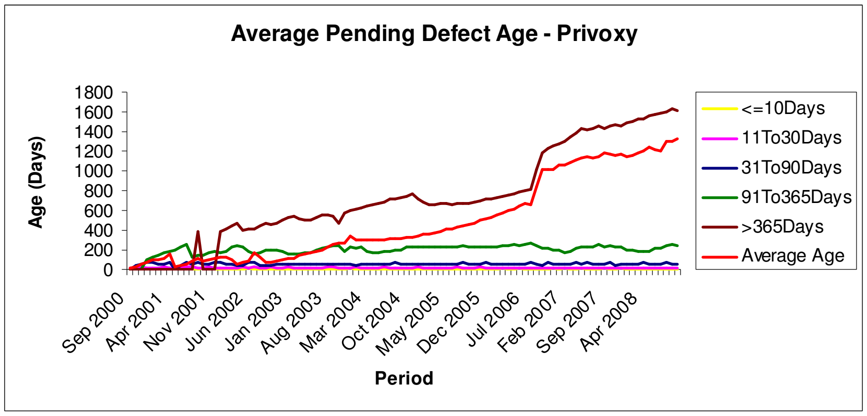

FIGURE 18: Defect Pending Age - Pending Age Wise (Privoxy)

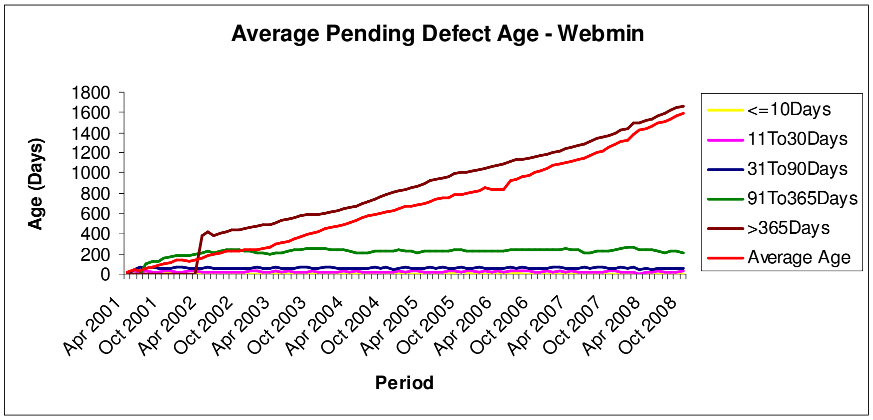

FIGURE 19: Defect Pending Age - Pending Age Wise (Webmin)

To observe the difference in pending age over the period for each of the 20 projects, ANOVA is applied. The statistics for two projects are highlighted in Table 7. Since the test statistic for both the projects is larger than the critical value, it is concluded that there is a (statistically) significant difference in average pending age over the periods. To analyze the overall defect pending age for all the selected projects together during the investigation period, average pending age for each of the 20 projects for various years is taken into consideration and standard analysis of variance (ANOVA) is applied which shows that there is significant change in defect pending age over the period (*F* (4,95)=15.2694; p<0.05; *F* critical=2.467494). The Table 8 also shows a continuous increasing trend in average defect pending age (days) for various years for all the 20 projects taken together.

Project	Period	Average Pending Age	Standard Deviation	ANOVA Results
Webmin	Jan.1, 2002 to Dec. 31, 2003	264.58	94.25	*F*(2,69)= 252.4181; p<0.05; *F* critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	647.36	108.64	
	Jan.1, 2006 to Dec. 31, 2007	1047.70	151.84	
Privoxy	Jan.1, 2002 to Dec. 31, 2003	151.14	74.96	*F*(2,69)= 163.3788; p<0.05; *F* critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	369.15	70.36	
	Jan.1, 2006 to Dec. 31, 2007	927.15	244.99	

TABLE 7: One Way ANOVA Statistics on Defect Pending Age

Period	Average Defect Pending Age (Days)	Standard Deviation
2004	286.51	174.13
2005	421.28	227.67
2006	593.92	288.80
2007	802.74	355.54
2008	897.11	364.85

TABLE 8: Average Defect Pending Age for 20 F/OSS Projects Together

4.9. Defect Resolution (Defect Type Wise)

An F/OSS user can submit defects in the form of bug reports, feature requests, patches or miscellaneous (translation, support requests, plug-ins, package requests or any other project specific category). As it is observed that there is generally an increasing trend in defect resolution age, hence further analysis is carried out to observe the resolution age of each of the defect type. Figure 20 and

21 show graphs for two of the selected projects where comparison is made between various defect types by distributing the resolved defects on the basis of defect resolution age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days).

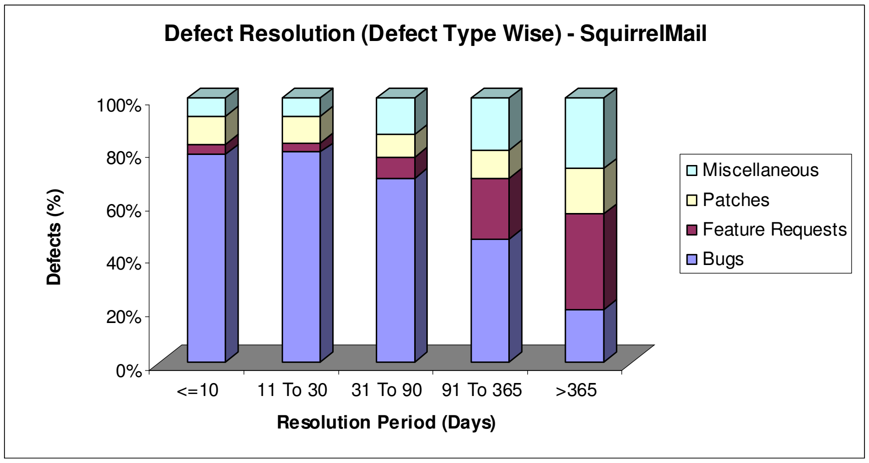

FIGURE 20: Defect Resolution - Defect Type Wise (SquirrelMail)

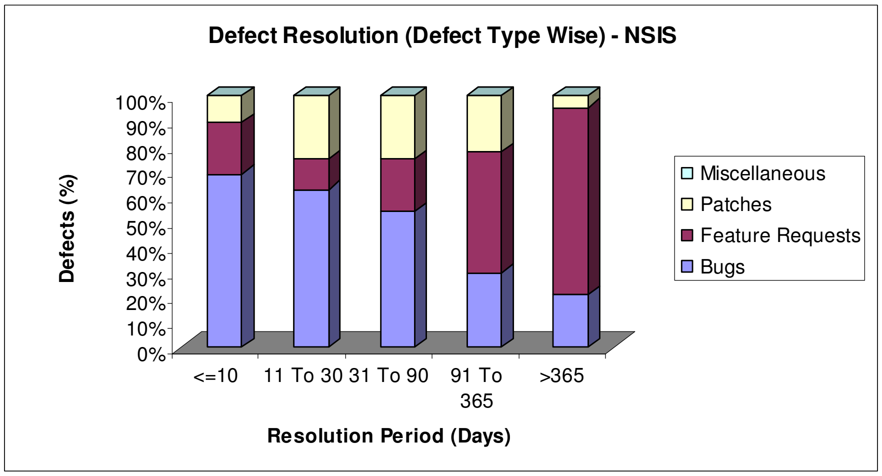

FIGURE 21: Defect Resolution - Defect Type Wise (NSIS)

It is found that all the defect types are dispersed among all the resolution age categories. It is also observed that proportion of bugs decrease with increasing resolution age while others (Feature Requests, Patches, Miscellaneous) increase with increasing resolution age. Further analysis of monthly average pending age is carried out in each of the defect type over the period (Figure 22 and 23). It is observed that each defect type is showing increasing trend in all the selected projects.

To analyze the defect pending age for each defect type taking all the selected projects together, average pending age in each defect type for each of the 20 projects for various years is taken into consideration and two way ANOVA is applied. The null hypothesis is that the differences between the defect types are consistent for various years. A significant year effect (*F*(4)=23.36133;p<0.05;*F* critical=2.395431) implies that there is a difference in the effect of different years on the defect pending age regardless of the type of defect. A significant defect type effect (*F*(3)=14.83437;p<0.05;*F* critical=2.628397) implies that there is a difference in the effect of different defect types on the defect pending age regardless of the level of year. While the interaction of year and Defect type (*F*(12)=0.748815;p>0.05; *F* critical=1.777693) implies that differences between the defect type are consistent for various years.

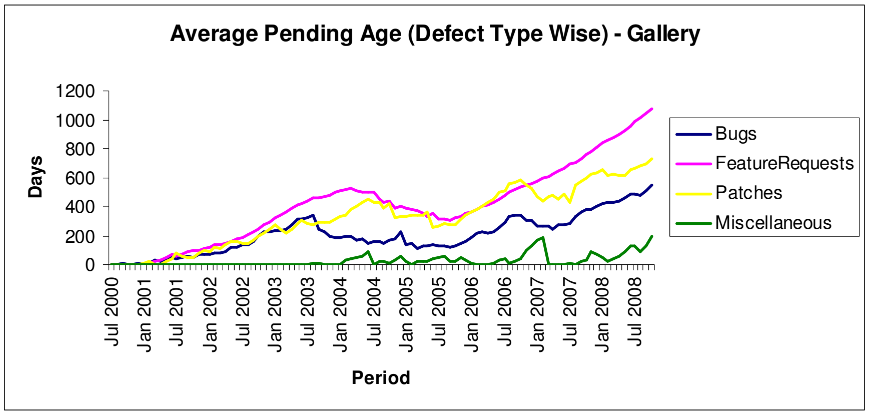

FIGURE 22: Average Pending Age - Defect Type Wise (Gallery)

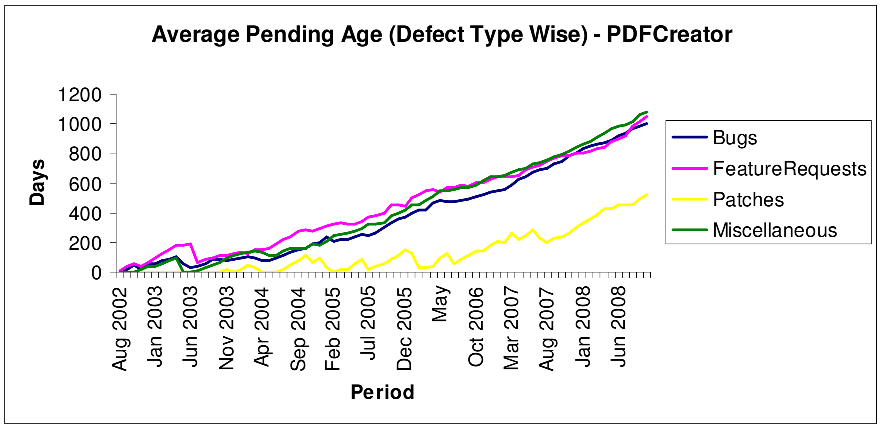

FIGURE 23: Average Pending Age - Defect Type Wise (PDFCreator)

5. PROBLEMS IN DEFECT MANAGEMENT

During the current study, various problems that have been identified in Defect Management are discussed as follows. Also an attempt is made to address these problems.

- It is observed that many F/OSS projects do not carry out defect resolution consistently and efficiently. The defect resolution is not able to keep pace with defect arrival thus accumulating pending defects. It is also found that backlog of pending defects accumulate gradually while their resolutions are carried out in bursty manner near the forthcoming releases. All these factors cause an increasing trend of overall resolution age as well as pending age. The detailed analysis shows that most of the defects are closed in reasonable time period while few defects take quite longer resolution time and aggravate the overall scenario. F/OSS development team should periodically review such long pending defects and prioritize them for resolution.

- It is also observed that there is no significant difference in resolution age of defects resolved with code fix or without any code fix (such as Duplicate, Out of Date, Won’t Fix, Works for Me, Invalid etc.). It is not justified that a defect is closed after 100 days or longer with the status information as Duplicate, Out of Date, Won’t Fix, Works for Me etc. Such behavior may cause loss of interest among participating users for further involvement. A process need to defined so that as soon as a defect is reported, members of development team should review it and if defect does not require any code change, it should be closed immediately with appropriate resolution status. By reducing Non-fix defect resolution age, overall resolution efficiency can be improved.

- It is found that all the defect types (Bugs, Feature Requests, Patches, Miscellaneous) are dispersed among all the resolution age categories although proportion of bugs decrease with increasing resolution age while others (Feature Requests, Patches, Miscellaneous) increase with increasing resolution age. It is also observed that each defect type is showing an overall increasing trend of pending age in all the selected projects. Ideally bugs should be resolved within shorter period depending upon the criticality of the bugs; while feature requests, patch submissions may be delayed till forthcoming releases/patches. Under miscellaneous category, the resolution should be carried out based upon the type of request. Due to volunteer nature of F/OSS participants, nobody can ensure that they will have enough time to respond to a defect quickly. So spreading the load across several development team members may lead to more reliability and to a shorter defect removal time.

- It has been found that in few F/OSS projects, the defect resolution status remains default (None) rather than being updated with relevant resolution status (Fixed, Duplicate, Out of Date, Won't Fix, Works for Me, Invalid etc.) even after the defect is closed. Although such defects are closed but the F/OSS users are not able to know exactly what actions have been taken on their reported defects. Defect Management System should have the functionality which enforces the development team to update the resolution status correctly while closing the defect.

- It has been found that in most of F/OSS projects, the F/OSS development team is not defining the priority of each defect being reported, although Defect Management System has the functionality to assign priority to reported defects. When a defect is reported, the priority is always set to default value 5 i.e. Normal (1-Highest, 9-Lowest) which is generally not updated by Development Team. Due to lack of prioritization of reported defects, the resolution of many critical defects may be delayed. F/OSS project development team should clearly define the criterion to identify the priority of each reported defect and make some of the team members responsible to assign the priority as per the criteria.

6. PROPOSED PROCESS FOR DEFECT MANAGEMENT

Based on the suggestions mentioned in the previous section, a process is proposed as shown in Figure 24, which can help to improve the effectiveness as well as efficiency of Defect Management.

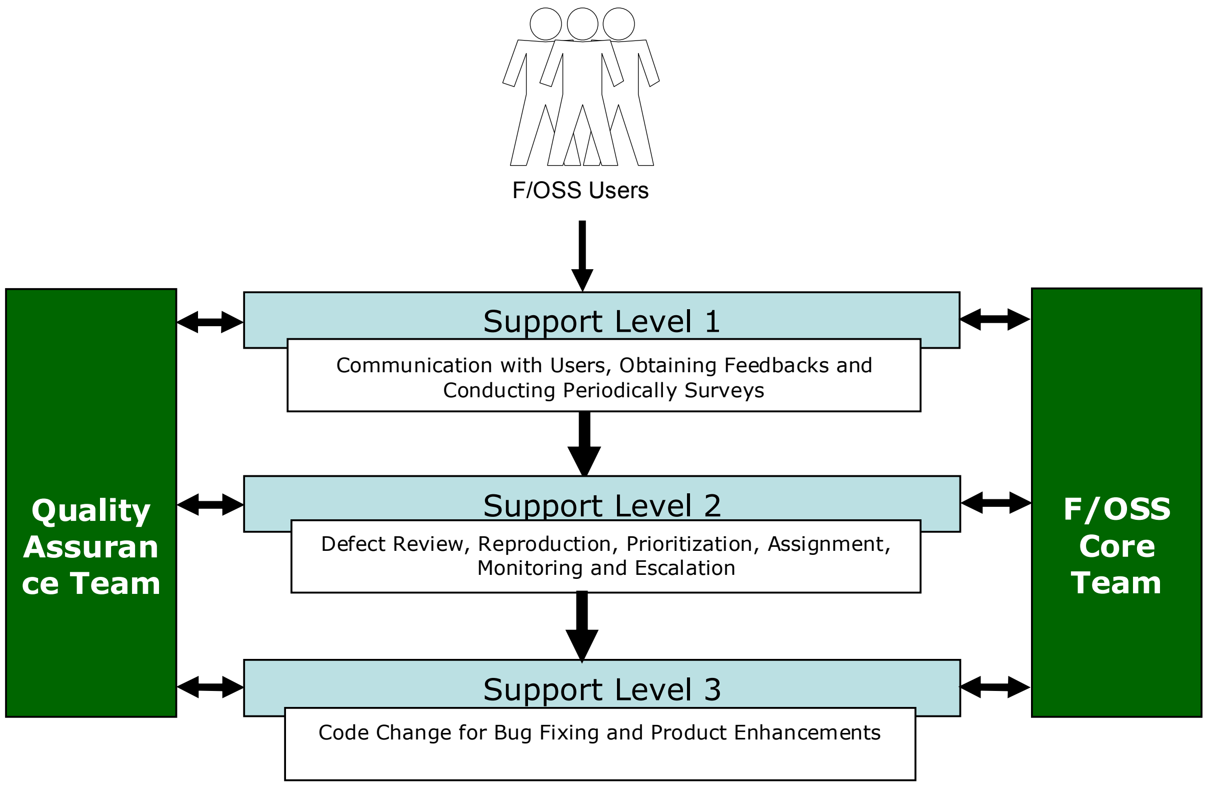
F/OSS Users
Support Level 1
Communication with Users, Obtaining Feedbacks and Conducting Periodically Surveys
Support Level 2
Defect Review, Reproduction, Prioritization, Assignment, Monitoring and Escalation
Support Level 3
Code Change for Bug Fixing and Product Enhancements
Quality Assurance Team
F/OSS Core Team

FIGURE 24: Proposed Process Diagram

It is proposed that support and maintenance activities should be distributed among various levels in order to improve the effectiveness and efficiency in Defect Management. The roles and responsibilities at these levels can be distributed as follows:

- **Support Level 1:** This level may comprise volunteer F/OSS users who may not have sufficient technical skill set to help development team but are ready to participate in F/OSS development process. This team should have responsibility to communicate with F/OSS users, obtain their feedback and conduct surveys periodically to know the level of satisfaction regarding usage of F/OSS Product and any issues that need to be addressed by development team.

- **Support Level 2:** This level may comprise volunteer F/OSS users cum developers who have sufficient technical skill set to help development team. They should be assigned the responsibility to review all the reported defects within stipulated period, make efforts to reproduce, collect additional information if required, set priority based upon prior defined criterion and assign them to team at level 3. They should keep on monitoring that no defect should remain pending for a long period without any appropriate reason. If there is any long pending defect without any justified reason, it should be escalated to Core Team for corrective measures. The members at this level should also resolve the defects which does not require any code change and set their appropriate status in the Defect Management System. They should also build knowledgebase comprising frequently occurring defects related to installation, configuration etc. and enabling F/OSS users to browse through easily.

- **Support Level 3:** This level may comprise volunteer F/OSS developers who have good technical skill set and knowledge of source code of F/OSS project. This team will have the responsibility to carry out necessary code changes to fix the defects as well to incorporate required feature enhancements. Whenever a defect is assigned, they should resolve the defect with in reasonable time frame. If some additional information is required about the defects, it should be obtained through level 2 team. Many times some of the defects can not be resolved due to constraints like software design, technology, resources, irreproducible etc. In all such cases, relevant information should be communicated to users timely.

- **Quality Assurance Team:** This team should comprise F/OSS volunteers preferably having some knowledge or experience in software quality assurance. They should have responsibility to monitor the activities carried out at all levels e.g. responsiveness towards users, defect resolution period, backlog of defects, code review etc. and should assure that quality is maintained at all the levels. They should generate and analyze the statistics periodically and should escalate serious concerns (if any) to core team.

- **F/OSS Core Team**: This team comprises the initiators and project leaders who have the overall responsibility. They should control the overall direction of project, take corrective measures for serious concerns and decide future strategy for forthcoming releases.

7. CONCLUSION

Defect Management Systems have been used to record and track defects for many years, but there is little analysis of the recorded defect data. Analyzing the defect data is of substantial value since it reveals how various variables connected to the defects change over time such as defect arrival rate, defect removal rate, defect resolution period, handling of pending defects etc. An analysis of more than 60,000 defect reports associated with 20 F/OSS projects reveals that many important insights can be gained through the analysis of defect data that has been recorded over the years. The quality of an F/OSS project can be improved a lot if defects are identified, reported and resolved in efficient manner. Generally an F/OSS project is developed by a small team of core developers which is surrounded by a community consisting of large number of globally distributed users. Not every F/OSS user has the technical skills to take part in code review or to carry out development. However, these users can contribute to the project by reporting bugs or by suggesting new features.

For effective Defect Management, the defect reports should be updated correctly and regularly. Also for efficient defect management, the defects should be resolved and closed at the earliest and consistently. During the analyses, it has been found that generally defect resolution is not performed consistently. This results in declining defect removal rate and an ever increasing average age of defect resolution. This problem needs to be addressed timely otherwise important user feedback is not incorporated into the software and many opportunities of improving the software are lost. It is also observed that defects get accumulated gradually and then additional efforts are put to resolve them near the forthcoming software releases. An observation of the BMI reports also confirms that backlog is increasing gradually but decreasing steeply. It is also found that a few defects remain pending for fairly long period of time in the Defect Management System. They are neither resolved nor their status is updated, if resolved. Such ignored defects keep on accumulating and result in increasing trend in overall defect pending age. The inefficient defect resolution has serious effects in the long term if effective countermeasures are not found. Moreover, as defects become older, reproducing them becomes increasingly more complex because the software continuously changes. Finally, users will perceive that their feedback does not have any impact and will stop providing valuable input. This minimizes the benefits that F/OSS projects can draw from peer review and user involvement, which is an important characteristic of F/OSS projects. A layered process is proposed where roles and responsibilities are clearly defined and distributed among F/OSS participants. F/OSS projects may use the proposed process which can help to improve the effectiveness and efficiency in Defect Management and thus assure better quality of F/OSS Products.

ACKNOWLEDGMENTS
We are thankful to the University of Notre Dame for providing access to Sourceforge Research Data Archive (SRDA) for retrieving data on F/OSS projects.

REFERENCES

1. Eric S. Raymond, "The Cathedral and the Bazaar", First Monday, 3(3), 1998.

2. Walt Scacchi, “Software Development Practices in Open Software Development Communities: A Comparative Case Study”, Proceedings of 1st Workshop on Open Source Software Engineering, May 2001, Toronto, Ontario, Canada.

3. Audris Mockus, Roy Fielding and James D. Herbsleb, “Two Case Studies of Open Source Software Development: Apache and Mozilla” ACM Transactions on Software Engineering and Methodology, 11(3): 309–346.

4. Dawid Weiss, “A Large Crawl and Quantitative Analysis Of Open Source Projects Hosted On Sourceforge”, Research Report ra-001/05(2005), Institute of Computing Science, Pozna University of Technology, Poland.

5. A. G. Koru and J. Tian, “Defect Handling in Medium and Large Open Source Projects”, IEEE Software, 21(4):54-61, July 2004.

6. Daniel German and Audris Mockus, “ Automating the Measurement of Open Source Projects”, Proceedings of the 3rd Workshop on Open Source Software Engineering, International Conference on Software Engineering, May 2003, Portland, Oregon, USA.

7. Stefan Koch, “Effort Modeling and Programmer Participation in Open Source Software Projects '', Information Economics and Policy, 20 (4): 345-355, 2008.

8. Ionic Stamelos, Lefteris Angelis, Apostolos Oikonomou and Georgios L. Bleris, “Code Quality Analysis in Open Source Software Development”, Information Systems Journal, 12(1): 43:60, 2002.

9. “SourceForge”, http://sourceforge.net/

10. Yongqin Gao, Matthew Van Antwerp, Scott Christley and Greg Madey, "A Research Collaboratory for Open Source Software Research", Proceedings of 29th International Conference on Software Engineering + Workshops (ICSE-ICSE Workshops 2007), International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007), May 2007, Minneapolis, Minnesota, USA.

11. Chiara Francalanci and Francesco Merlo, “Empirical Analysis of the Bug Fixing Process in Open Source Projects”, Open Source Development, Communities and Quality, Springer Boston, 275 :187-196, 2008.

12. Martin Michlmayr and Anthony Senyard, “A Statistical Analysis of Defects in Debian and Strategies for Improving Quality in Free Software Projects”, The Economics of Open Source Software Development, Elsevier B.V., 2006, pp 131–148.

13. Stephen H. Kan, “Metrics and Models in Software Quality Engineering”, Second Edition, 2003, Pearson Education.