American Journal of Software Engineering and Applications
2013; 2(2) : 68-79
Published online April 2, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20130202.16

A metric based approach for analysis of software development processes in open source environment

Parminder Kaur, Hardeep Singh

Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar-143005, India

Email address:
parminderkaur@yahoo.com (P. Kaur), hardeep_gndu@rediffmail.com (H. Singh)

To cite this article:
Parminder Kaur, Hardeep Singh. A Metric Based Approach for Analysis of Software Development Processes in Open Source Environment, *American Journal of Software Engineering and Applications*. Vol. 2, No. 2, 2013, pp. 68-79. doi: 10.11648/j.ajsea.20130202.16

Abstract: Open source software (OSS) is a software program whose source code is available to anyone under a license which gives them freedom to run the program, to study, modify and redistribute the copies of original or modified program. Its objective is to encourage the involvement in the form of improvement, modification and distribution of the licensed work. OSS proved itself highly suited, both as a software product and as a development methodology. The main challenge in the open source software development (OSSD) is to collect and extract data. This paper presents various aspects of open source software community, role of different types of users as well as developers. A metric-based approach for analysis of software development processes in open source environment is suggested and validated through a case study by studying the various development processes undertaken by developers for about fifty different open – source software's.

Keywords: Free Software, Freedom, Open Source Software (Oss), Propriety Software, Oss Developer Community, Oss Metrics

1. Introduction

Free software (Stallman, 1983) provides users the freedom to run, copy, distribute, study, change and improve the software. Four essential freedoms are provided to user's i.e.

• The freedom to run any of the programs for any purpose (freedom 0) i.e. the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of purpose, without communicating the developer or any other specific entity..

• The freedom to study how the program works, and how it does computing according to users wish (freedom 1). Access to the source code is a precondition for this.

• The freedom to redistribute copies so user can help their neighbors (freedom 2) i.e. it must include binary or executable forms of the programs as well as source code, for both modified and unmodified versions.

• The freedom to distribute copies of modified versions to others (freedom 3). By doing this, user can give a chance to benefit from his/her changes to whole community.

In order for these freedoms to be real, they must be permanent and irrevocable as long as nothing does wrong. If the developer of the software has the power to revoke the license, or retroactively change its terms, software is not free.

Open source software (Raymond, 1998) is software for which the source code is available to everyone for modification and inspection. This is in contrast with propriety software which cannot be inspected and modified by anyone. In the last ten years, open source software (OSS) has attracted the attention of not only the practitioner, but also the business and the research communities. OSS has proven to produce software of high quality, functionality and wide development. Open Source Definition include the GNU Public License (GPL), the BSD license used with Berkeley Unix derivatives, the X Consortium license for the X Window System, and the Mozilla Public License. Open source software definition includes the following terms listed as:

• Free Redistribution: -Anyone who received the software legally can share all of it with anyone without additional payments.

• Source Code: -The program must include source code, and must allow distribution in source code as well as compiled form. Intermediate forms such as the output of a preprocessor or translator are not allowed. The source code of the software must be distributed as well or be available at reasonable reproduction cost.

• Derived Works:-The modification of the software and the distribution of this derived work must be allowed.

- Integrity of the Author's Source Code:-The distribution of modified source code must be allowed although restrictions to ensure the possibility to distinguish the original source code from the derived work are tolerated, e.g. requirement of different names.
- No Discrimination against Persons or Groups: -The license must not discriminate against any person or group of persons.
- No Discrimination against Fields of Endeavor: -The license must not forbid the usage of the software in specific field of endeavor, e.g. business or genetic research.
- Distribution of License:-The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
- License must not be specific to a Product:-The rights given by the license must not be different for the original distribution and any other one even when it takes place in a totally different context.
- License must not contaminate other software:- The license must not demand any condition on the software distributed along with the licensed software.

2. Related Work

Open source software is now the demand of era. Literature survey includes various aspects of different researchers. Open source is a term that has recently gained currency as a way to describe the tradition of open standards, shared source code, and collaborative development behind software such as the Linux and FreeBSD operating systems, the Apache Web server, the Perl, Tool Command Language (Tcl), and Python languages, and much of the Internet infrastructure, including Bind (the Berkeley Internet Name Daemon servers that run the Domain Name System), the Sendmail mail server (sendmail.org), and many other programs.

(John, 1998) has viewed that qualitative data analysis (QDA) is a symphony based on three notes: Noticing, Collecting, and Thinking about interesting things. The process has characteristics such as Iterative and Progressive, Recursive. QDA has simple foundation but the process of doing qualitative data analysis is complex. (Katherine and Tony, 2002) has viewed as open source has been most successful in back-end types of applications such as operating systems. They analyzed the projects listed on the www.freshmeat.net developer forum on the basis of two indicators i.e. vitality on the project and the popularity of the project. Vitality has been calculated using the number of announcements about a project and the time since its last release. Popularity is based on the number of people who subscribe to the project.

(Fredrik, 2002) has asked that open source development model is not only producing software but also produces the interacting system of knowing, learning and doing, which organizes the community and its relations with other communities. Users are allowed to download the software from the Internet and use it without charge and granted the right to study the software's source code, to modify the software, and to distribute modified or unmodified versions to others.

(Jin and Madey, 2004) has told that OSS community as a complex, self-organizing system. Developers are main components in the network. An OSS developer community is composed of a group of loosely-connected contributors. An OSS community can be classified as different roles: active and passive users, peripheral developer, central developer, core developer, project leader. Data is gathered from the 2003 data dump provided by SourceForge.

(Jin Xu et al, 2005) has included users and developers in research paper. Passive users download code and use it for their needs. Active users discover and report bugs, suggest new features. The Peripheral developers irregularly and Central developers regularly fix bugs, add features, submit patches, provide support, write documents and exchange other information. Core developers manage CVS releases and coordinate peripheral developers and central developers. Project leaders guide the vision and direction of a project. Many difficulties exist in collecting, cleaning, screening and interpreting data. (Chris, 2005) has used models for Apache, Mozilla, and Net-Beans to show the relationships between tools, agents, their nonfunctional requirements and functional requirements. The quality assurance (QA) process can be modeled in Mozilla Web browser as a rich hypermedia, Apache release process with flow graph, and Net-Beans requirements and release process, can be modeled formally and reenact. The approaches to modeling software development processes within and across these communities, as well as issues and trade-offs that arise along the way are described.

(David et al, 2006) has suggested that essential characteristics of the software like reliability, maintainability or sustainability cannot be identified by source code inspection alone, but have to include the environment in which it has been created. The requirements are structured into the aspects of various functional, non functional, technical, organizational, legal, economical and political.

(Ismail et al, 2007) has given information about open projects that can be obtained from two main sources, either the source code or the document produced. This is a static source that enables analysis on the quality of the product. Dynamic information is needed to go through the stages of development and the communication between the peers. Various metrics are used to quantify the roles of core developers, release stability are used.

(Henrike et al, 2009) has proved that data collection is time consuming process and requires some effort. To solve this problem, tools are developed for metrics analysis of a large number of software projects. Measurement and data collection is performed in three phases, two automated and one manual phase.

(David, 2011) tells us about the recent analysis of companies contributing code to the Linux kernel. It shows that large companies including Novell, IBM, Intel, Nokia and Texas Instruments are getting serious about engaging in community development. Organisations such as the Linux Mobile (LiMo) Foundation are encouraging their members to work with community projects “upstream”, that is, with the community rather than in isolation, to avoid missing out on millions of dollars worth of “unleveraged potential” (PDF link). Sun Microsystems and AOL are prominent examples of companies which went full speed into community development, but were challenged (to say the least) in cultivating a mutually beneficial relationship with community developers.

3. Open Source Software Community

Open source software community (figure 1) can be classified into two groups as: User group and Developer group.

User group further categorizes in Passive users and Active users. Passive users have no contribution in the development of the software projects. They are attracted to OSS mainly due to its high quality and potential of being changed when needed. Active users not only use the system, but also try to understand how the system works by reading the source code. They can suggest new features, discover and report bugs and exchange other useful information by posting messages to forums and mailing lists.

Peripheral Developers contribute occasionally new functionality or features to the existing system. They irregularly fix bugs, provide support, write documents and exchange information. Their period of involvement is short and sporadic. Central Developers are the major development force of OSS systems. They regularly fix bugs, add new features, submit patches, provide support, write documents and exchange information. Core Developers are responsible for guiding and coordinating the development of an OSS project. Core Developers are those people who have been involved with the project for a relative long time and have made significant contributions to the development and evolution of the system. OSS projects in which single Project Leader no longer exists and the Core Developers form a council to take the responsibility of guiding the development, such as the Apache Group and the PostgreSQL core group.

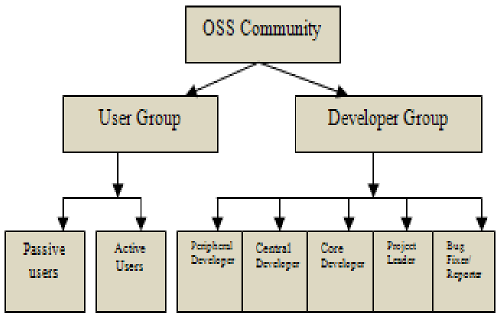


Figure 1. *Open source software community [5].*

Project Leader is often the person who has initiated the project. He or she is responsible for the vision and overall direction of the project. Bug Fixers fix bugs that either they discover by themselves or are reported by other members. Bug Reporters discover and report bugs. They do not fix the bugs themselves, and may not read source code either. They assume the same role as testers of the traditional software development model. The existence of many Bug Reporters assures the high quality of OSS.

Contributors communicate with each other through online tools and platforms. OSS development process is open involving a large number of developers submitting contributions that may have significant variations in quality. The communication tools are Concurrent Version System CVS), mailing list, bug tracking systems, online discussions forums.

4. Sourceforge.Net: an Open Source Software Community

SourceForge.net is the world's largest Open Source software development web site. On July 2011, the SourceForge repository hosts more than 300,000 projects and has more than 2 million registered users. The aim of SourceForge.net is to enrich the Open Source community by providing a centralized place for Open Source developers to control and manage Open Source software development. To fulfill this mission goal, it offers a variety of services to projects that are hosted, and to the Open Source community. SourceForge.net stores a set of common attributes for all projects. These attributes are divided into two groups, the first contains static information about the project such as the license, and the second contains information such as the number of code changes committed to Concurrent Version System Active users. Passive Users have no direct contribution in the development of the software projects. They are attracted to OSS mainly due to its high quality and the potential of being changed when needed. Active users not only use the system, but also try to understand how the system works by (CVS).

SourceForge.net uses relational databases to store project management activity and statistics (sourceforge.net Research Data available at www.nd.edu). There are over 100 relations (tables) in the data dumps provided to university of Notre Dame. Some of the data have been removed for security and privacy reasons. SourceForge.net cleanses the data of personal information and strips out all OSTG (Open Source Technology Group) specific and site functionality specific information. On a monthly basis, a complete dump of the databases (minus the data dropped for privacy and security reasons) is shared with Notre Dame. The Notre Dame Researchers have built a data warehouse comprised of these monthly dumps, with each stored in a separate schema. Thus, each monthly dump is a snapshot of the status of all the SourceForge.net projects at that point in time. As of March 2007, the data warehouse was almost 500 GBytes in size, and is growing at about 25 GBytes per month. Much of the data is duplicated among the monthly dumps but changes in project activity and structure can be discovered by comparing data from the monthly dumps. To help researchers determine what data is available, an ER-diagram and the definitions of tables and views in the data warehouse are provided. However, SourceForge.net site provide hints as to what types of data might be available in the SourceForge.net data warehouse to support research into the Free/Open Source Software.

A. Types of Data that Can be Extracted from the SourceForge.net Research Data Archive

The following are types of data that have been extracted from the SourceForge.net Research Data Archive:

- Project sizes over time (number of developers as a function of time presented as a frequency distribution)
- Development participation on projects (number of projects individual developers participate on presented as a frequency distribution).
- The extended- community size around each project including project developers plus registered members who participated in any way on a project (discussion forum posting, bug report, patch submission, etc.)
- Date of project creation (at SourceForge.net)
- Date of first software release for a project
- SourceForge.net ranking of projects at various times
- Activity statistics on projects at various times
- Number of projects in various software categories, e.g., games, communications, database, security, etc.

Since all of the archived data is stored in a relational database, data to support F/OSS investigations will be extracted using SQL queries against the data warehouse.

5. Open Source Software Metrics

Metrics are used for measurement, comparison or to track performance or production. Metrics helps to compare the performance of software at various levels. Metrics helps in good decision making or improvements in the project. Metrics should be accurate, timely and actionable. Metrics are derived from the earlier data. These must be understandable, economical and must be useful at the various levels of the development of the project (Pinker S., 2009), (David et al, 2003), (Kaur and Singh, 2011), (Kaur & Kaur, 2012). Metrics used in this research work are:-

A. Total Number of Contributions

A large number of users contribute towards the development of the project but developers appear to become more influential contributors to their open source. Developers play more roles in open source projects. The contributors can contribute either to a single project or multiple projects. This metric is related to the number of contributors for a given project irrespective of their affiliations to other projects.

B. Average Domain Experience of Contributors

Contributors participating in the software development have some expertise and experience in that domain area contribute to the project. This metric helps in evaluating the average experience of all the contributors taken together and can be represented as

Cumulative Experience of Contributor i.e.

$$E = \sum_{i=1}^{n} e_i$$

[Where e_i is the experience of an individual contributor in that domain.]

Average Experience of Contributors i.e. $E_{avg} = E/N$

[Where N is total number of contributors.]

C. Average Time for a Completion of a version of Project

The average time for a completion of a version of project can be calculated as:

Average Time i.e. Tavg = $T_{total} \ / \ N_{version}$

Where T_{total} is total time taken to develop all the versions and $N_{version}$ is the total number of versions generated. Greater average would indicate software development processes resulting from various factors like low number of contributors, their lack in experience or complexity of the project etc.

D. Bugs Track per Version

The detected bugs could be allocated to various development processes like requirement specification, design, coding and testing. This metric helps in measuring the total number of bugs located and repoted per version. Greater the number of bugs detected/tracked more is the inefficiency of various development processes.

E. Patch Accept Ratio

Patches are change sets that can be applied to the software using a specific tool: the patch tool. Patches may introduce new features, fix bugs, translate strings to a different language, or re-structure parts of the software. Every contributor sends a patch to the developer mailing list for the enhancement of the product. Patch tool takes a patch file that contains the details of the modifications and applies them to the original version of some code in order to create a new, updated version.

Patch Accept Ratio i.e. total no of patches accepted /total no of patches sent

A high Patch Accept Ratio indicates high competence of contributors and reverse is true for the less patch ratios.

F. Total Number of Weekly Downloads

Weekly downloads of a particular software can be obtained from sourceforge.net website.

6. Automation of Open-Source Software Metrics

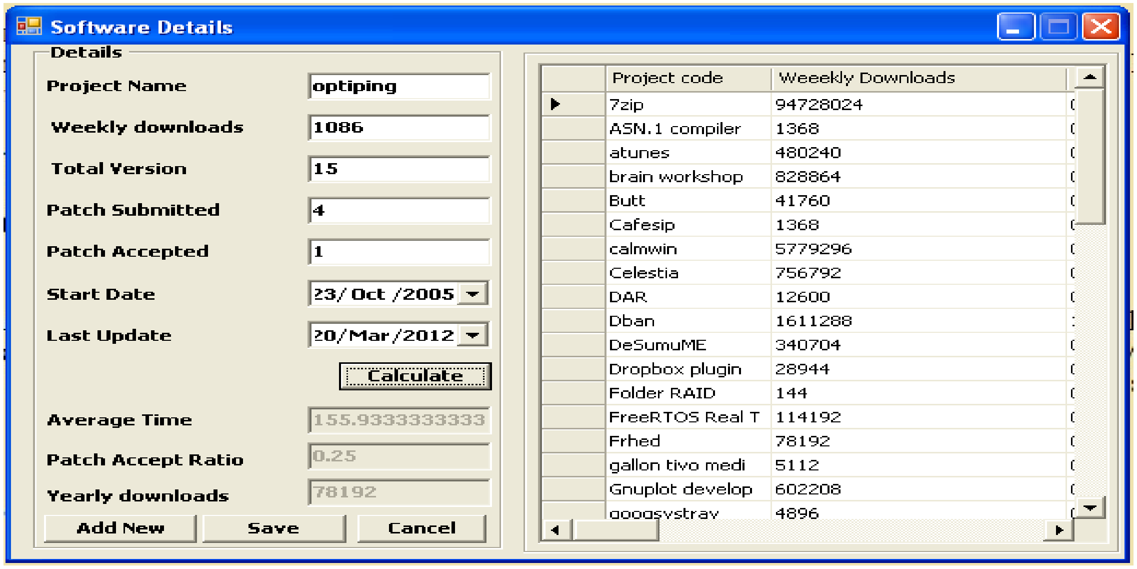
To automate the open-source software metrics, suggested in section VI, a case study deals with 50 open source software's (obtained from www.sourceforge.net) is taken (Annexure 1). A tool named as Software Metric calculation Tool, is developed using Asp.Net Framework with MS-Access as database storage. This study is performed on the basis of following parameters to critically analyze the behavior of open-source software community:

- No. of Contributors
- Average domain experience of Contributors
- Weekly downloads
- Patch submitted
- Patch accepted
- Time to generate first version
- Time to generate last update
- Total number of versions
- Total number of bugs/fixations/updations w.r.t. a particular version
- Weekly downloads

A. Working of Software Metrics Calculation Tool

The working of Software Metrics Calculation tool is shown as below:

- Figure 2 shows that details of software such as project name, weekly downloads, total versions, patch submitted, patch accepted, start date and last update date of the project. Average time and patch accept ratio are calculated automatically by clicking on the calculate button. Information about the new software is added in the database by selecting add new option. Average time is calculated by taking the diff between start date and last update divided by number of versions.


Software Details

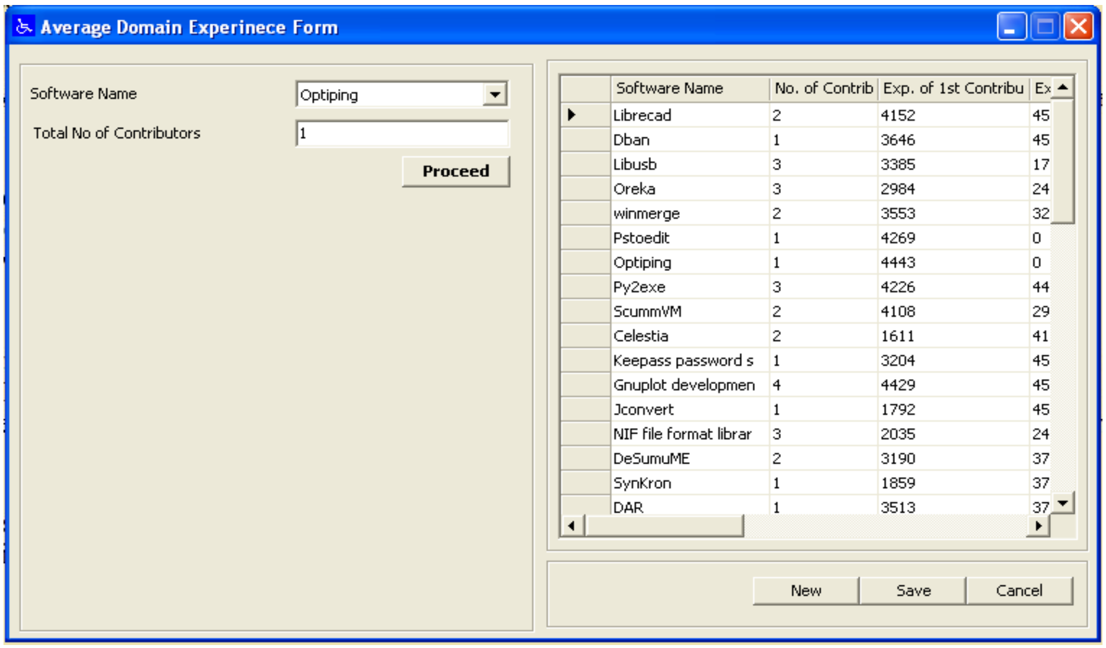
Details

Project Name	optiping
Weekly downloads	1086
Total Version	15
Patch Submitted	4
Patch Accepted	1
Start Date	23/ Oct /2005
Last Update	20/Mar/2012
Average Time	155.9333333333
Patch Accept Ratio	0.25
Yearly downloads	78192

Project code	Weeekly Downloads
7zip	94728024
ASN.1 compiler	1368
atunes	480240
brain workshop	828864
Butt	41760
Cafesip	1368
calmwin	5779296
Celestia	756792
DAR	12600
Dban	1611288
DeSumuME	340704
Dropbox plugin	28944
Folder RAID	144
FreeRTOS Real T	114192
Frhed	78192
gallon tivo medi	5112
Gnuplot develop	602208
googsystray	4896

Figure 2. Software Details Form.

- Figure 3 show that user first selects the name of the software and then specify the number of contributors involved in the software.

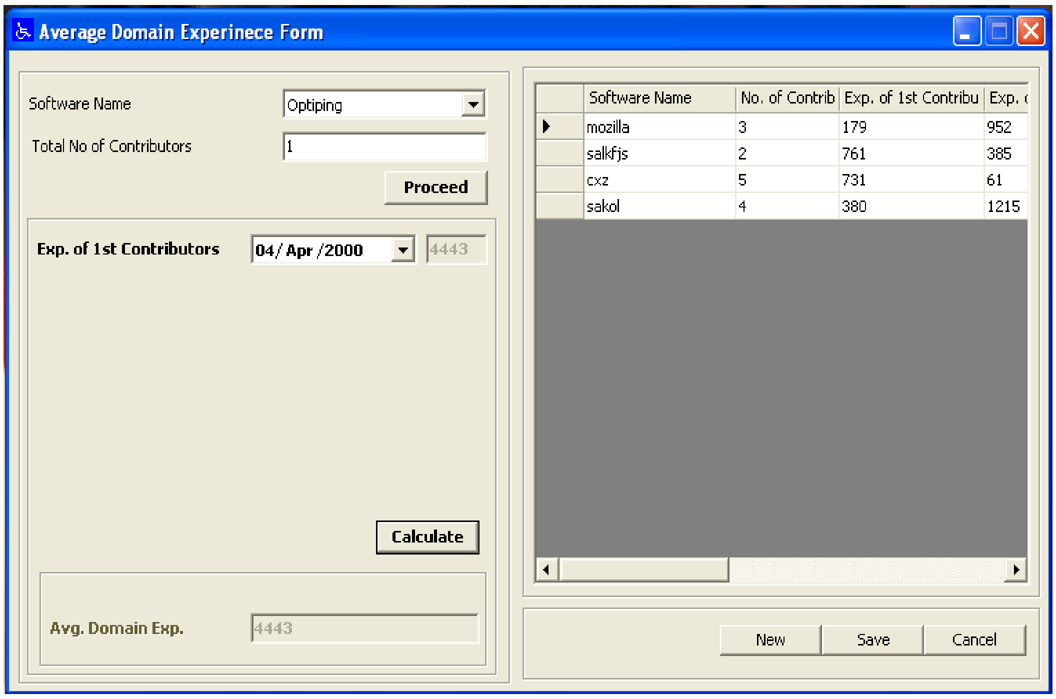

Average Domain Experinece Form

Software Name	Optiping
Total No of Contributors	1

Software Name	No. of Contrib	Exp. of 1st Contribu	Ex
Librecad	2	4152	45
Dban	1	3646	45
Libusb	3	3385	17
Oreka	3	2984	24
winmerge	2	3553	32
Pstoedit	1	4269	0
Optiping	1	4443	0
Py2exe	3	4226	44
ScummVM	2	4108	29
Celestia	2	1611	41
Keepass password s	1	3204	45
Gnuplot developmen	4	4429	45
Jconvert	1	1792	45
NIF file format librar	3	2035	24
DeSumuME	2	3190	37
SynKron	1	1859	37
DAR	1	3513	37

Figure 3. Number of Contributors form

- Figure 4 shows average domain experience, which is calculated in days. If more than one contributor is involved in particular software then experience of each contributor is calculated in number of days and sum is divided by the total number of contributors.

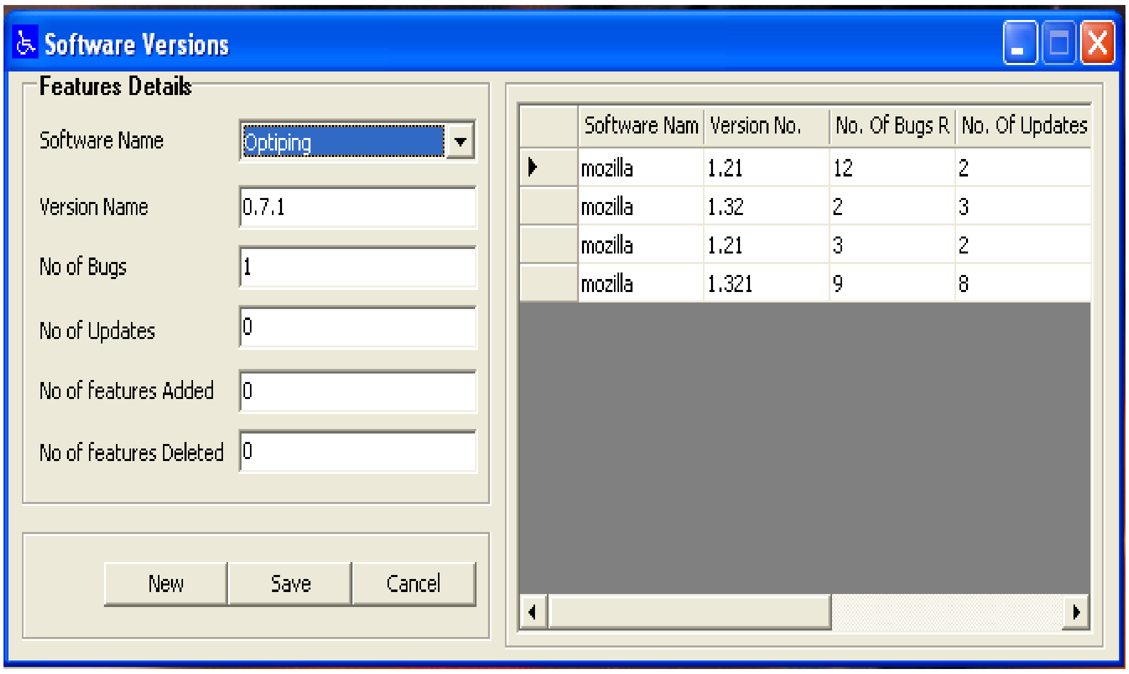

Average Domain Experinece Form

Software Name	Optiping
Total No of Contributors	1
Exp. of 1st Contributors	04/ Apr /2000 (4443)
Avg. Domain Exp.	4443

	Software Name	No. of Contrib	Exp. of 1st Contribu	Exp. (
▶	mozilla	3	179	952
	salkfjs	2	761	385
	cxz	5	731	61
	sakol	4	380	1215

Figure 4. *Average Domain Experience Form*

- Figure 5 represents the way how details of each version are stored in database. These details includes software name, version name, number of bug fixes, number of existing features updated, number of new features added and number of existing features dropped.


Software Versions

Features Details

Software Name	Optiping
Version Name	0.7.1
No of Bugs	1
No of Updates	0
No of features Added	0
No of features Deleted	0

	Software Nam	Version No.	No. Of Bugs R	No. Of Updates
▶	mozilla	1.21	12	2
	mozilla	1.32	2	3
	mozilla	1.21	3	2
	mozilla	1.321	9	8

Figure 5. *Versions Details Form*

- Five types of graphs are generated to show the development process of all the versions, total number of bugs fixed, total number of features update, total number of features added and deleted. Figure 6 represents shows development graph of all the softwares.

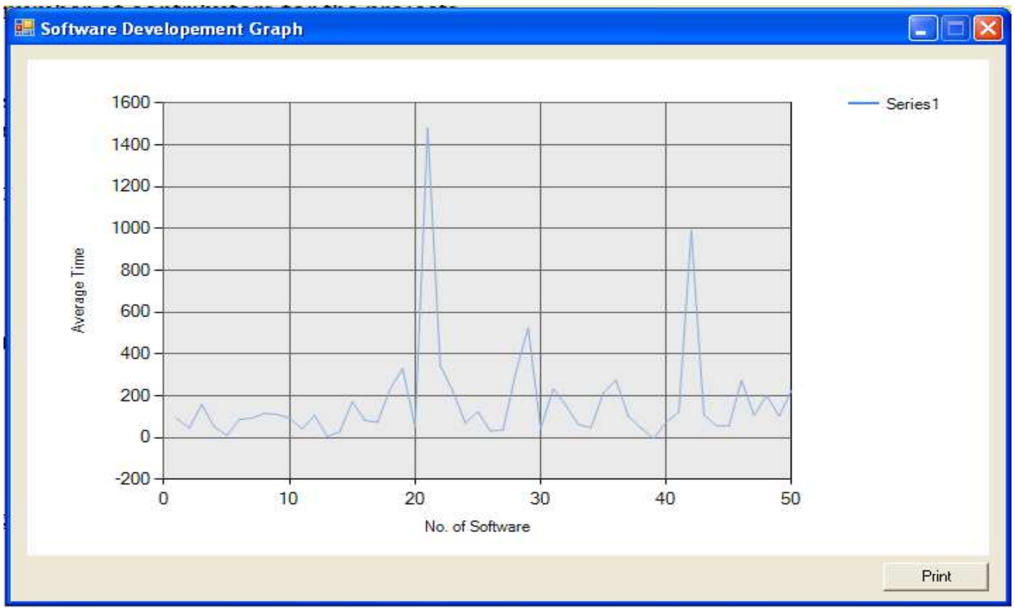


Figure 6.**-** *Development Graph Form*

• Around 50 softwares have been taken for validating the proposed set of metrics. Every ach software is evaluated on the basis of total number of bug's fixed, total number of features update, total number of features added and deleted. All details with respect to softwares are manually collected from www.sourceforge.net data dump. A We obtained password is obtained in order to access their data dump. However, the proposed metrics are being calculated automatically by the Software Metric Calculation Tool. Figures 7, 8, 9, 10 show all these activities simultaneously with respect to software WinMerge (a Windows-based tool for visual difference display and merging, for both files and directories), having weekly downloads numbering 34126, with 72 total number of versions in 2011 having first version on 2000. The total number of patches submitted for this product is 3016, out of this; only 2182 are utilized for generating next versions (Annexure 1).

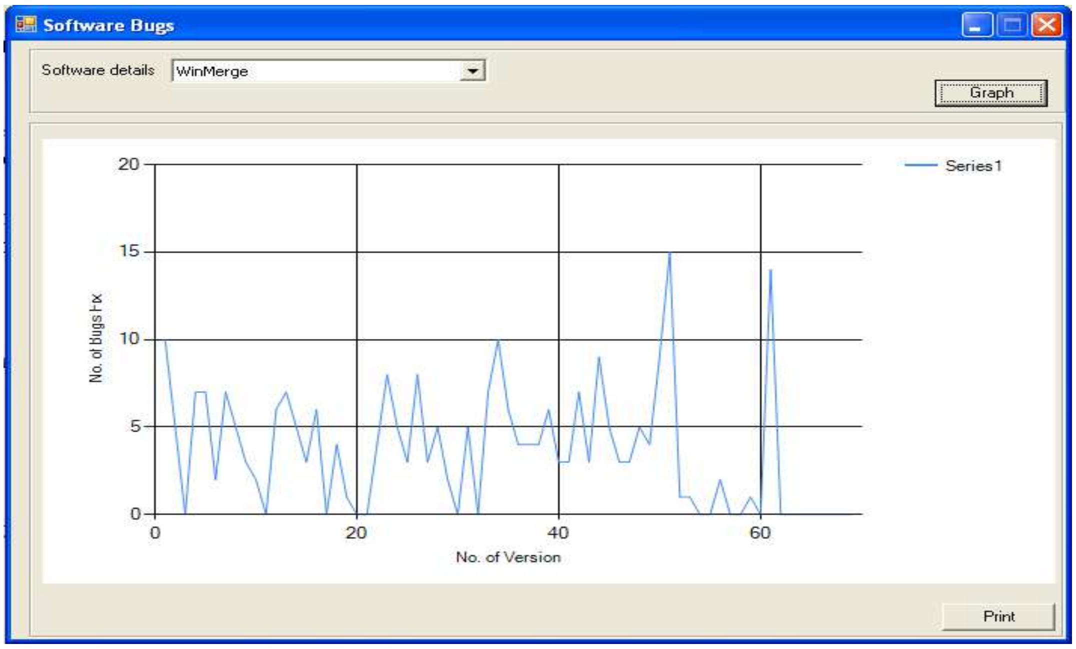

Software details: WinMerge

Figure 7. *-This shows graph of bug fixes performed in WinMerge. Maximum bugs are fixed in older versions.*

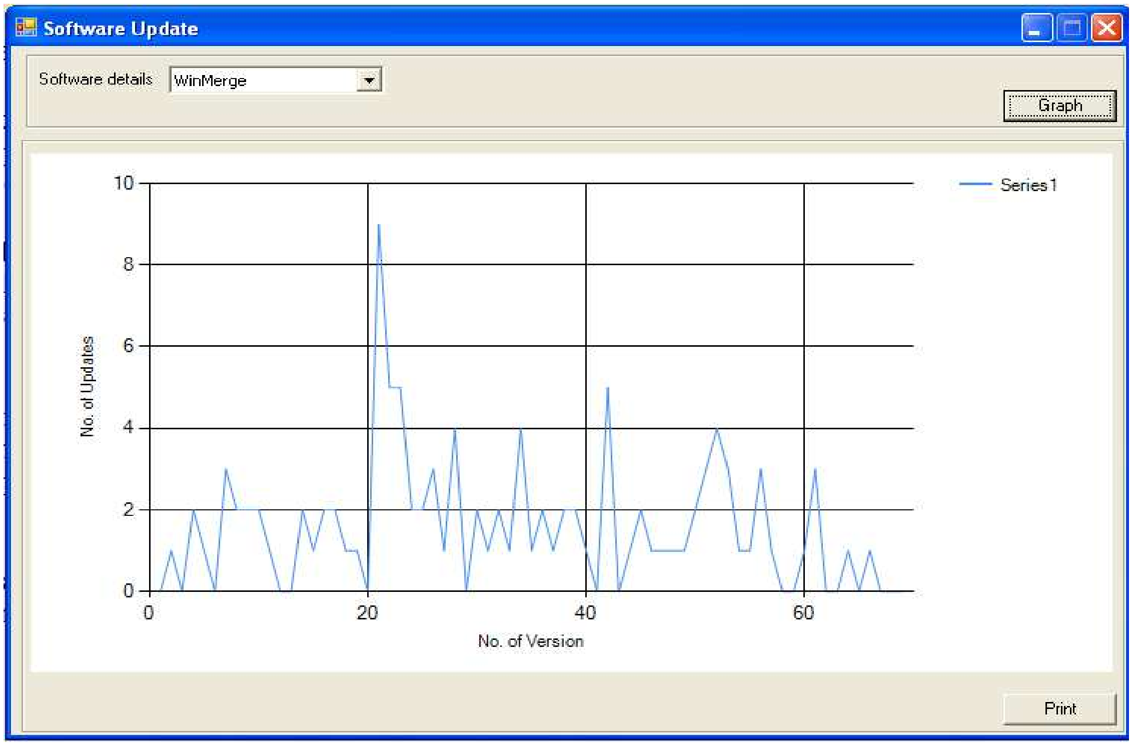


Figure 8.- *This figure represents updates performed in WinMerge.*

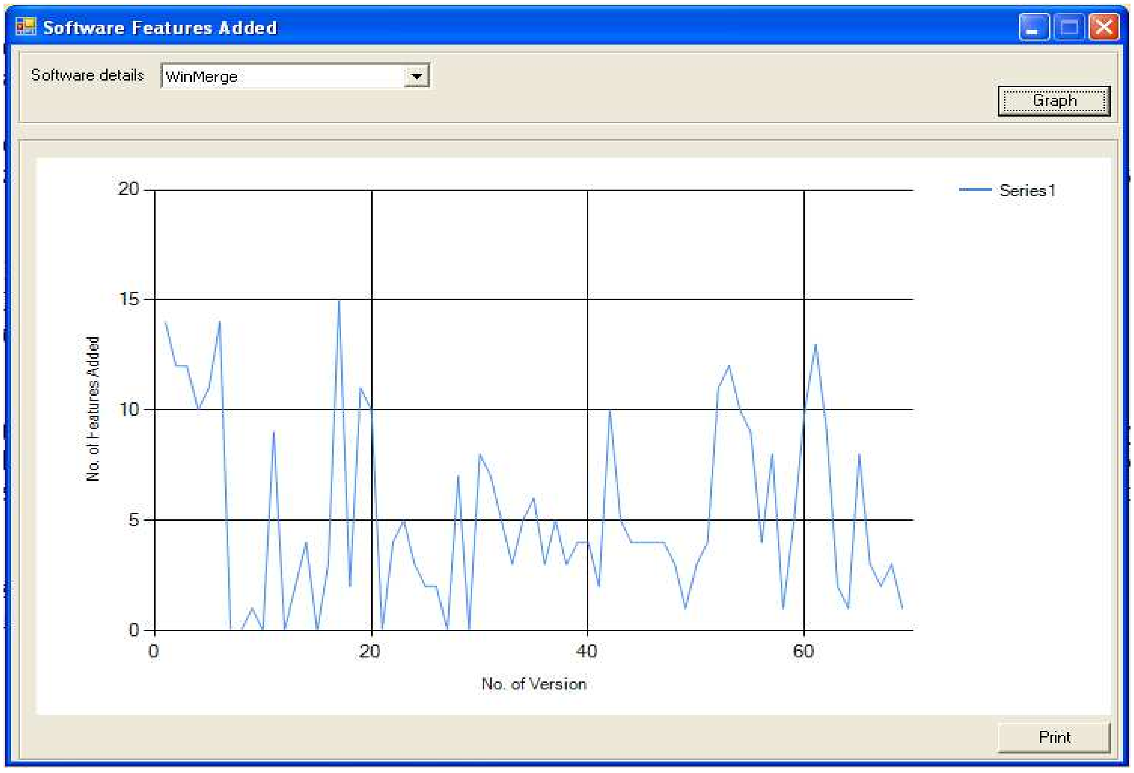


Figure 9.- *This figure shows graph of new features added to WinMerge.*

7. Conclusions

This paper makes an effort to explore some concepts related to free software, open source software and open source software community. Open source software community is a combination of active users as well as passive users which consists of developers, project leaders, bug fixers & reporters. A great data is available with respect to open source software community through which one can make the comparative analysis between different above said issues and concerns. A metric-based approach is explained to check the evolution of open source software development processes. Future work will include the automation of collection of raw data from various resources.

Annexure - 1

S No.	Project Name	Weekly Downloads	Contributors Name	Staring time in the domain	Patch Submitted	Patch Accepted	Time to generate first versions (Start)	Time till last version complete (last update)	Total number of versions
1	Warkeys	27423	Warkeys	19-2-2006	-	-	20-2-2006	12-1-2012	42(124)
2	Calmwin free antivirus	80268	Alch, gianlivigi tiesi	22-3-2004, 5-6-2000	10	-	25-3-2004	3-4-2012	62
3	Celestia	10511	Ajtribick , chris laurel	5-1-2008, 23-2-2001	7	-	23-2-2001	29-12-2011	4
4	7-zip	1315667	Igor paul	17-8-2000	459	20	10-11-2000	30-6-2011	62
5	Winprint	206	Mieczyslaw nalewaj, przemek czerkas	12-11-2001, 12-4-2004	-	-	12-4-2004	22-5-2012	2(9)
6	Winmerge	34126	Christian list, dean grimm	11-9-2002, 28-8-2003	3016	2182	20-10-2000	14-11-2011	72
7	The ASN.1 compiler	49	Lev walkin	11-11-2000	14	-	6-3-2004	16-6-2011	9(54)
8	T38 modem	149	Jordan kojouharov, vyacheslav frolov	2-11-2005, 27-11-2003	4	1	2-11-2005	5-7-2011	6(19)
9	Out2Gcal	4	Thisita	14-7-2010	-	-	14-7-2010	29-9-2010	7
10	Oreka	268	Bruce kingsland, henrih, Ralph atallah	2-4-2004, 17-10-2005, 11-11-2008	3	-	17-10-2005	18-10-2011	8
11	OptiPNG	1086	Cosmintruta, rctruta	4-4-2000,	4	1	23-10-2005	20-3-2012	15(30)
12	NIF file format library and tools	4606	Alphex, amorilia, pacific morrowind	9-11-2006, 8-9-2005, 6-6-2009	32	7	25-5-2009	20-2-2012	179
13	LMS desktop	4	Gianni ven hoecke, marten, Patrick law werts	23-12-2009, 18-12-2009, 6-5-2010	6	-	5-6-2010	14-10-2011	18
14	Libusb-win32	8279	Stephan meyer, travis robinson, xiaofan chen	26-2-2003, 22-7-2007, 18-2-2006	15	1	5-4-2003	23-1-2012	30
15	Libjpej turbo	958	D.R.commander	13-8-2004	34	1	4-2-2010	10-2-2012	10
16	Lame	23189	Alexen Derliedinge, Gabriel bouvigne, Robert hagemem,takehiro tomingo	16-10-2000, 27-11-1999, 29-11-1999, 28-11-1999	60	14	17-11-1999	28-2-2012	21
17	Keepass password safe	131359	Dominik reichl	28-8-2003	77	62	15-11-2003	14-5-2012	73
18	Jaris FLVPlayer	132	JGM, YLM	27-6-2007, 6-3-2010	1	-	18-5-2008	29-8-2011	11
19	Hylafax	3	Stavan jardine	31-1-2001	-	-	17-2-2006	6-4-2011	20
20	Hava Fun	5	Elfman	21-8-2009	1	-	22-8-2009	28-11-2011	3
21	Grsync	121	Piero clrsoni	3-2-2010	3	1	3-2-2010	13-1-2012	5
22	Graphics magick	757	Bob friesenhaHn	30-12-2000	27	-	7-2-2003	28-4-2012	15(46)

23	Googsystray	68	Jim duchek	11-1-2000	3	1	8-9-2009	6-6-2011	15(24)
24	Gnuplot development	8364	Hons-bernhard broeker, clark gaylord, lars hecking, ethan merritt	20-4-2000, 31-1-2000, 27-4-2000, 2-6-2001	607	344	31-1-2000	12-3-2012	18
25	Gallon tivo media server	71	John kohl, leon nicholls	31-7-2004, 7-8-2003	11	7	11-12-2004	24-3-2011	8
26	Frhed	1086	Kimmo varies	18-10-2002	38	4	10-8-2008	6-6-2011	14(15)
27	Folder RAID	2	Liran	1-7-2010	-	-	31-8-2010	3-12-2011	9
28	Dropbox plugin	402	July ighor	5-11-2009	-	-	5-11-2009	2-10-2011	8
29	DeSmuME	4732	Guillaume duhamel, zeromus	11-9-2003, 18-3-2002	138	46	23-12-2006	26-4-2012	16
30	Dban	22379	Darik horn	10-6-2002	2	2	6-9-2002	29-6-2011	16
31	WCD- change directory for DOS and Unix	6	Erwin waterland	9-9-2001	58	41	9-9-2001	29-2-2012	31(68)
32	libreCad	2576	Ries van twisk, dongxi li	20-1-2001, 10-3-2011	35	2	13-8-2010	24-4-2012	6
33	Vantage	7	Mchansy, raoul van bugen	23-7-2009, 15-5-2011	-	-	27-4-2009	5-9-2011	5
34	Ultradefrag	15580	Dmitri arbhangelski, gearspec, justin dearin, zsolt nagy, Stefan pendl	25-6-2007, 28-12-2008, 8-7-2001, 11-3-2008, 28-4-2009	-	-	25-6-2007	22-4-2012	57(58)
35	Pstoedit	1424	Wolfgang glunz	25-9-2000	2	2			
36	Im4java	158	Bernhard bablok	1-1-2001	-	-	23-1-2009	4-4-2012	11
37	DAR	175	Denis corbin	23-10-2002	30	3	25-10-2002	15-4-2012	47
38	Butt	580	Daniel nothen	20-9-2007	-	-	21-9-2007	14-10-2011	16
39	Open video player	713	AdamGreen Baren, Charles newman, dan sparacice, james mutton, Nicholas brooking, pankaj , tommy petrovic	19-10-2009, 22-10-2008, 2-7-2010, 22-10-2008, 11-11-2008, 24-5-2010, 4-6-2010	3	-	21-8-2008	29-6-2011	23
40	Atunes	6670	Fleax	27-6-2006	42	18	7-3-2006	30-4-2012	49
41	Cafesip	19	Amit chatterjee, becky Mc	16-6-2004, 23-6-2004	-	-	14-5-2005	26-4-2012	16
42	FreeRTOS Real Time Kernel	1586	Richard	4-6-2004	59	2	9-6-2004	14-5-2012	65
43	SynKron	2592	Matus tomlein	4-5-2007	-	-	7-5-2007	25-6-2011	13
44	Py2exe	3479	Jimmy retzlaff, Mark hammOnd, Thomas heller	7-11-2000, 16-2-2000, 3-2-2000	27	5	29-11-2000	3-10-2011	17
45	Kdiff3	3994	Joachim eibl	25-7-2002	13	1	25-7-2002	14-10-2011	11(32)

46	Nbuexplorer	3847	Petrusek	23-8-2007	-	-	2-10-2009	26-5-2012	26(27)
47	Jconvert	636	Eds	10-7-2007	-	-	17-7-2008	8-5-2011	11
48	Brain workshop	11512	Paul hoskison	8-8-2008	1	-	8-8-2008	5-4-2012	12
49	ScummVM	19912	Eugene Sandulenke, Strangerke	5-3-2001, 10-6-2004	1536	1282	5-3-2001	21-2-2012	37
50	iramuteq	123	Pierre	4-1-2010	-	-	4-1-2010	16-1-2012	9

References

[1] Anas Tawileh, Omer Rana, and Steve McIntosh (2008), " A social networking approach to F/OSS quality assessment", In Proceedings of the First international conference on Computer-Mediated Social Networking (ICCMSN'08), Maryam Purvis and Bastin Roy Savarimuthu (Eds.). Springer-Verlag, Berlin, Heidelberg, 157-170. DOI=10.1007/978-3-642-02276-0_16 http://dx.doi.org/10.1007/978-3-642-02276-0_16

[2] C. Jensen and W. Scacchi (2005), "Process Modeling Across the Web Information Infrastructure", Wiley InterScience, available at http://www.ics.uci.edu/~wscacchi/Papers/New/Jensen-Scacchi-SPIP-ProSim04.pdf

[3] Cruz, T.Wieland and A. Ziegler (2006)," Evaluation Criteria for Free/Open Source SoftwareProducts Based on Project Analysis", Wiley InterScience. available athttp://www.idi.ntnu.no/grupper/su/courses/tdt10/curricula2010/P5-1-Cruz06.pdf

[4] David P., Waterman A., Arora S. (2003)," The free/ libre & open source software survey for 2003", STANFORD UNIVERSITY, CALIFORNIA, USA, available at http://www-siepr.stanford.edu/programs/OpenSoftware_David/FLOSS-US-Report.pdf

[5] David Neary, (2011), " Open Source Community Building: A Guide to getting it Right", available at http://www.visionmobile.com/blog/2011/01/open-source-community-building-a-guide-to-getting-it-right/

[6] Fredrik Hallberg (2002), "The use of the open source development model in other than software industries. http://www.opensource-marketing.net/OSD.pdf

[7] H. Barkmann, R. Lincke and W. Lowe (2009), "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects", advanced Information Networking and Applications Workshops. http://www.arisa.se/files/BLL-09.pdf

[8] Ismail Ari (2007), "Quantitative Analysis of Open Source Software Projects", the Handbook of Computer Networks, Volume 2, http://users.soe.ucsc.edu/~ari/ari-quan-OSS.pdf

[9] J. Seidel (1998)," Qualitative Data Analysis", SAGE Publications., available at http://www.quarc.de/fileadmin/downloads/Qualitative%20Data%20Analysis_the%20N-C-T%20Modell.pdf

[10] J Xu and G. Madey (2004)," Exploration of the Open Source Software Community", Available athttp://www.cse.nd.edu/~oss/Papers/naacsos04Xu.pdf

[11] J. Xu, Y. Gao, S. Christley and G. Madey (2005), "A topological analysis of the open source software development community", In proceedings of 38th Hawaii International Conference on Systems Science, Hawaii,http://www.nd.edu/~oss/Papers/7_11_07.PDF

[12] K. Stewart and T. Ammeter (2002), "An exploratory study of factors influencing the level of vitalityand popularity of open source projects", In proceedings of international conference on information systems, available at: http://www.rhsmith.umd.edu/faculty/kstewart/ResearchInfo/StewartAmmeter.pdf

[13] Kaur P. and Singh H. (2011)," Measurement of Processes in Open Source Software Development",in proceedings of Journal , Trends In Information Management, (TRIM) University of Kashmir, Srinagar, ISSN-0973-4163, Volume 7, Issue 2, pp 198-207, available athttp://www.inflibnet.ac.in/ojs/index.php/TRIM/article/viewFile/1254/1135

[14] Kaur M. and Kaur P. (2012), "A Review of an Open Source Software Community", In proceedings of National Conference in Emerging Computer Technologies (CECT 2012), vol. 2, page 107-110.

[15] Nicolas Ducheneaut(2005), "Socialization in an Open Source Software

[16] Community: A Socio-Technical Analysis", Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA, 94304, USA, available at http://www2.parc.com/csl/members/nicolas/documents/JCSCW-OSS.pdf

[17] Pinker S. (2009),"Software Estimation, Measurement, and Metrics", GSAM Version 3.0, available athttp://www.stsc.hill.af.mil/resources/tech_docs/gsam3/chap13.pdf

[18] Raj Agnihotri, Murali Shanker and Prabakar Kothandaraman, "Theorization of the open source software phenomenon: a complex adaptive system approach", Journal of Management and Marketing Research, available at https://docs.google.com/viewer?a=v&pid=gmail&attid=0.2&thid=1354765351e41826&mt=application/pdf

[19] Stallman, Richard M (2010), "Free Software, Free Society",Selected Essays of Richard M. Stallman, Second Edition. Boston, Massachusetts: GNU Press. ISBN 978-0-9831592-0-9.

[20] Scott Christley, Jin Xu, Yongqin Gao, Greg Madey, (2006), "Public goods theory of the open source development community using agent-based simulation", Computer Science and Engineering, University of Notre Dame., available at https://docs.google.com/viewer?a=v&pid=gmail&attid=0.5&thid=1354765351e41826&mt=application

[21] Vinay Tiwari (2011), "Software Engineering Issues in Development Models of Open Source Software",International Journal of Computer Science and Technology, Vol. 2. http://www.ijcst.com/vol22/1/vinay.pdf

[22] Walt Scacchi (2011)," Understanding the Requirements for Developing Open SourceSoftware Systems", publication with revisions.http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-OS-Requirements.pdf