iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VII **Month of publication:** July 2021

DOI: https://doi.org/10.22214/ijraset.2021.36983

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VII July 2021- Available at www.ijraset.com

Calculating and Understanding the Usefulness of JIRA Developers Communities

Aayush Rai[1], S. P. S. Chauhan[2]
[1, 2]*School of Computer Science and Engineering, Galgotias University, Greater Noida, India*

Abstract: ***Tools for project management and bugs or problems following have become useful for managing the event method of the Open supply package. These tools clear the communications task between developers and make certain the quantifiability of the given project. The additional data developers square measure ready to interchange, the brighter square measure the achievements, and therefore the larger is that the limit of developers keen on a change. During this paper, we tend to gift an introductory determined investigation of the companies-structure of developers in JIRA by learning some well-liked comes hosted within the repository. We tend to study however, these firms perform in terms of the given issue's issue-resolution time. The most contributions of this work square measure the authentication of the survival of firms in developer networks, and therefore, the actual finding that the given issue's problem resolution time isn't correlative with the importance of a developer company.***

I. INTRODUCTION

Over the past 10 years, analysts are finding the usefulness of the communities of Open Source software, and now a days, Open Source systems aren't any expand deliberated to be the kids of a lesser God. Such systems reaching the position is famous and appreciated [9]. Fast net connections, tablets, smartphones, and gadgets perpetually connected to the web facilitate ASCII text file developers to remain in-tuned with one another to provide software package. Communication processes area unit the most part for ASCII text file ideal development, and tools able to control the communication among a gaggle of individuals developing and making one thing along area unit thus substantive. The communication facet is in the middle and is that the key to properly control the event method. The data of a project must be well on the market for the event team throughout the event method. Once a brand new developer adds in the development team, the higher the communication method works, the quicker the new developer may become useful and also the learning curve will be diminished. In this study, we have a tendency to be evaluated the developers' structure for seven comes introduced in JIRA1, a proprietary issue pursuit product developed by Atlassian. JIRA adds issue pursuit, bug pursuit, and project management functions, and involves tools permitting movement from challenger. In keeping with Atlassian. It is used for issue pursuit and project management by over twenty five thousands customers round the world. An Issue Tracking System (ITS) may be a repository employed by software package developers as facilitate for the software package development method. It helps restorative maintenance activities like Bug pursuit systems, together with alternative varieties of requests of maintenance. Since JIRA is changing into additional and additional well-known between developers, it's useful to know the impact of such a product on the structure of developer communities. Numerous studies have indicated that developers area unit concerned within the method of making new ASCII text file software package that area unit organized which a transparent structure exists. Linux, as an example, isn't the results of a disordered method dead by disordered developers. Their success in developing a really difficult system, used even by NASA3, wasn't merely supported luck. However, this doesn't mean that ASCII text file ideals perpetually result in the event of quality product. Queries like the subsequent arise concerning open supply communities: "Can I trust one thing made by individuals operating for complimentary, driven solely by dedication and happiness?" [11] "Can software package developed while not an ad arrange and hard deadlines be of great feature?" "How will developers work expeditiously while not central project coordination? Diseconomies of scale will have an effect on the communication method of a gaggle of developers operating along. Once newcomers be part of a project, the structure becomes more hard to control, and it's going to be difficult to stay track of World Health Organization is doing what. Tools like JIRA, facilitate to decrease coordination issues, to extend the extent of communication, and to rescale the project by decreasing time of release. The major goal of this analysis was to supply sensible proof showing whether or not developer's area unit organized with outlined teams/structures and if such groups perform otherwise in terms of productivity. We have a tendency to outline the work rate of a team because of the average fixing time for any given issue.

We have a tendency to answer the subsequent analysis questions: RQ1: Will the ASCII text file software package developer's network graph contain communities? Open supply comes hosted in JIRA have communities. Results show that the seven ASCII text file comes evaluated have numerous communities, starting from eight to sixteen. RQ2: area unit there variations in fixing time between communities? Whereas, JIRA Developer Communities have completely different average issue fixing times, the distribution of issue varieties and priority is analogous across the communities. The rest of the paper is structured as follows: in Section II, we have a tendency to offer an outline of the connected works. In Section III, we have a tendency to make a case for the dataset we have a tendency to be used and also the methodology. We have a tendency to conclude with Section V and VI by analyzing the threats to validity and summarizing our finding

II. CONNECTED WORK

The bazaar is one in every of the foremost used unorthodox comment to outline the development of Linux, and numerous studies have used the "Cathedral-Bazaar" figure [18] to outline the resources of a (differently) systematised approach to development [2] [3] [5] [6] [13]. The structure of such separated development been intensely checked by several analysts. Small world development and scale-free actions area unit found within the supply Forge development network by Xu et al [25], considering 2 developers well-connected if they join forces within the same project. [23] assembled sensible social network information from blogs, email lists, and websites, to create models of development accustomed fake however users joined and left comes.

Ehrlich et al [10] used social network research to be told however people in international software package development groups sight and gain information. Madey et al. [15] checked structural information on over thirty-nine thousand open supply comes hosted at SourceForge.net together with over thirty three thousand developers and concepted that open supply software package development might be designed as self-organizing, Project sizes, developer project participation, and clusters of connected developers area unit checked Crowston et al. [7] checked a hundred and twenty project groups from SourceForge, discovering that open supply development groups dissent in their centralization of a communication, from comes focused on one developer to comes extremely separated and viewing a distributed arrangement of communication between developers and active users. Larger groups tend to possess additional separated patterns of communication.

Different analysts [16] checked the shape of the developer partnership with the network of a developer to examine failures at the file level. Different studies have checked the shape of the ASCII text file software package communities to elucidate social aspects in development. Steinmacher et al [21] [22], known twenty studies providing sensible proof of obstructions faced by beginners to OSS comes whereas conducive.

They analyzed fifteen totally different barriers, that we have a tendency to classified into 5 categories: social interaction, beginners past information, finding some way to begin, documentation, and technical hurdles. The authors additionally classify the issues regarding their origin: beginners, community, or product. Zhou et al. [27] found, exploitation issue pursuit information of Mozilla and Gnome, that the chance for a beginner to be an extended Term Contributor is related along with their enthusiasm and atmosphere. Shah [19] found the inspirations of attendants from 2 software package development communities and finds that the majority attendants area unit impressed by either a requirement to use the software package or programming enjoyment. The latter cluster, enthusiasts or hobbyists, area unit crucial to the large relevance and property of open source software package code: they wrestle assignments that may go unfinished, area unit for the most part "need-neutral" as they create selections, Associate in Nursing specific an ambition to keep up the clarity, compatibility and elegance of the code. The aim of hobbyists develop over-time; most join in the community as a result of they have need for the software package and keep as a result of they get pleasure from programming within the selected community.

Ortu et al. [17] studied fourteen ASCII text file software package comes developed exploitation the Agile board of the JIRA repository. They checked all the comments hooked up by the developers concerned within the comes and that we analyse whether or not the courtesy of the comments affected by the quantity of developers concerned over the years and therefore the time needed to mend any given issue.

Results indicated that the amount of courtesy within the communication method among developers will have an effect on each the time needed to mend problems and therefore the attraction of the project to the additional polite developers were, the less time it took to mend a problem, and, within the majority of the checked cases, the additional the developers wished to be a part of the project, the additional they were willing to continue performing on the project over time. during this paper, we have a tendency to do a additional common analysis by mensuration and describing communities of developers at intervals JIRA.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VII July 2021- Available at www.ijraset.com

III. EXPERIMENTAL SETUP

A. Dataset

We have a tendency to created our information set by taking data from the Apache software package Foundation Issue Resolving System, JIRA4. We extracted the ITS of the Apache Software Package Foundation by taking problems from 2002 to Dec 2013.Table I shows the gathering of the seven comes finalized for our research, displaying variety of comments documented for every project and also the number of developers concerned. We have a tendency to electoral comes with the most variety of comments.

Table I: selected projects statistics

Project	# of comments	#of developers
HBase	91017	952
Hadoop Common	61959	1244
Derby	52669	676
Lucene Core	50153	1108
Hadoop HDFS	42209	851
Hive	39003	758
Hadoop Map-Reduce	34794	876

B. Developer's Network

We tend to open-up the developer network supported the information enclosed in JIRA. JIRA grants users to post problems (with a bunch of properties like maintenance sort, priority, etc.) and to investigate them. We tend to create developer network modeling nodes, that symbolize developers, and edges from node A to node B, that symbolize once developer A was commenting on developer B's issue. During this manner, we tend to nonheritate a directed network. We tend to used Gephi5 [1] to see the nonheritable network. Gephi could be a collective visual image and exploration tool. We tend to run the modularity algorithmic program, supported the algorithms developed by Blonde [4] and Lambiotte [14], to accomplish the network communities. Figure one shows Associate in Nursing example of the network graph we tend to nonheritable.

Fig. 1: example of developer's graph extracted from Lucene- Core Project

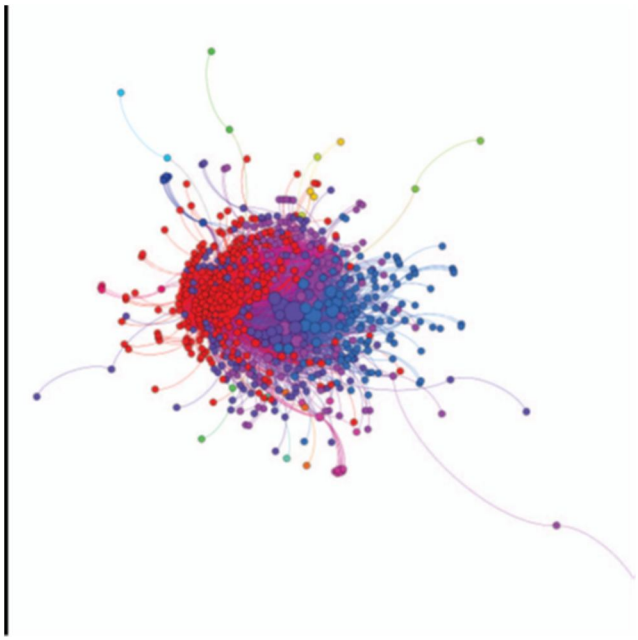


Fig. 2: example of developer's graph extracted from Derby project

IV. RESULT AND DISCUSSION

A. *Will the ASCII text file Code Developer's Network Graph Contain Communities?*

Motivation. Understanding developer structure in open source comes permits each work groups and management to own higher management over the total project. Each issue sorting and employment schedule could take pleasure in having a transparent read of however the developers square measure organized and the way productivity is unfold across work groups. For this reason, our 1st analysis question aims to explore the presence of communities in JIRA developer networks.

Approach. We have a tendency to made the developer network graph as delineated in III-B. Every Node represents a developer UN agency posted/commented on a problem, and every Edge represents a developer UN agency commented on another developer's issue. We have a tendency to applied the modularity algorithmic rule [14] to get developer communities.

TABLE II: selected projects network statistics

Project	Modularity	Avg. Degree	Avg. Clustering Coeff.	# of Communities
HBase	0.284	5.553	0.458	9
Hadoop Common	0.335	5.618	0.295	16
Derby	0.194	5.171	0.483	12
Lucene Core	0.268	3.494	0.338	15
Hadoop Map-Reduce	0.333	5.041	0.241	13
Hive	0.334	4.359	0.286	17
Hadoop HDFS	0.285	5.276	0.310	9

Findings. Open Source comes in JIRA do have communities. Table II shows the network metrics for the analyzed comes. The Modularity metric represents the perimeters fraction that fall inside the given teams minus the expected such fraction if edges were divided willy-nilly. It is effective if the amount of edges inside teams surpasses the amount predicted on the idea of likelihood. The Average Degree denotes the average node degree (as the total of out-degree and in-degree). The Average agglomeration Coefficient metric determines on the average however shut node neighbours are to being a lot (complete graph) [24]. The column in the last represents the communities' amount found by the formula [14]. Output shows that the seven ASCII text file comes analyzed have variety of communities, starting from eight to sixteen.

B. Is there Variation in Time of fixing Between Work Teams?

Motivation. Support the findings associated with the primary analysis objection, we all knew of the communities presence in JIRA developer networks. It's conjointly of affection to research these communities so as to grasp if and the way they work is divided. These type of knowledge are often helpful throughout problem triaging and planning of work division. For instance, if we discover a communities partitioning a higher share of problems of maintenance, then it's seemingly that a issue of maintenance are allotted to them, or if there's a community with a quick average issue resolution time, then this community are often allotted bug problems once the discharge date is close. Approach. We tend to use the developers' network obtained within the initial analysis question. For every project, we tend to analyze its developer communities by evaluating the typical issue resolution time, the amount of problems resolved, and also the distribution of maintenance sort and priority of mounted problems. Findings. Whereas JIRA Developer Communities have totally different average issue fixing times, the division of issue varieties and priority is analogous across the communities. From tables III to IX show, for every project, the amount of developers happiness to a specific community, the amount of problems resolved, and also the average issue fixing time. The first result's associated with Pareto's law (20% of developers doing eightieth of the problem resolution) [12]. There are solely a number of communities taking care of the bulk of problems. These communities have a unique average resolution time, and this variety is freelance from the size of community and from the amount of mounted problems.

Table III: Derby communities' statistics

Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
6	3	1	5.7
4	21	185	159.5
13	7	2	176
3	246	2918	214.8
2	110	878	216.4
1	144	2497	242.2
12	4	2	258.3
0	91	371	260.9

Table IV: Hadoop Common communities' statistics

Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
9	3	1	0.8
10	6	1	10.8
6	27	220	52.4
3	142	2358	71.9
2	503	2161	86.2
0	294	2991	120.8
1	61	206	122.3
4	85	214	158.7
5	32	362	193.8

We judged the Pearson's coefficient of correlation among the common issue fixing time and also the range of problems resolved, and among the common issue fixing time and also the community size. We have a tendency to find a rather weakend correlation (<0.3) with the sole exception being for Hadoop Map/Reduce. This output shows that the common issue resolution time may be a property of a developer community, and it doesn't rely on the community size. There's an excellent distinction within the range of problems solved by the communities and also the average issue resolution time per community. This truth is consistent across all of the seven comes analyzed. So as to know however the productivity is distributed across the developer communities, we have a tendency to find, for every community, the mounted issue distribution of priority and maintenance sort. Figures three and four shows, for every project, the distribution of maintenance sort and also the priority of the problems mounted by a community. For every project, the bar-chart on the left represents the distribution of mounted issue maintenance sort, and also on the left, the bar-chart represents the distribution of mounted issue priority. For pretty much all the communities, these 2 distributions area unit are same; specifically, there aren't any experienced communities, as an example, communities determination principally Bugs with important priority. We will then conclude that the common issue resolution time is property of the community and doesn't rely on the community size, the amount of mounted problems, or the upkeep sort and priority.

Table V: Hadoop HDFS communities' statistics

Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
4	3	1	10.8
6	114	1622	45.2
2	198	1804	67
1	147	529	218.6
0	235	596	317.8

Table VI: Hadoop Map/Reduce communities' statistics

Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
5	109	520	66.7
2	153	1863	70.9
7	58	85	119.6
0	114	413	128.4
1	168	1048	152.2
3	208	447	471.1

Table VII: HBase communities' statistics

Community Id	Community Size	# of Fixed Issues	AVG Fixing time [Days]
0	61	516	45.8
2	154	1628	47.5
6	42	377	60.6
5	90	5787	64.2
1	145	1493	69.9
3	287	3214	85.1
4	162	1706	90.3

Table VIII: Hive communities' statistics

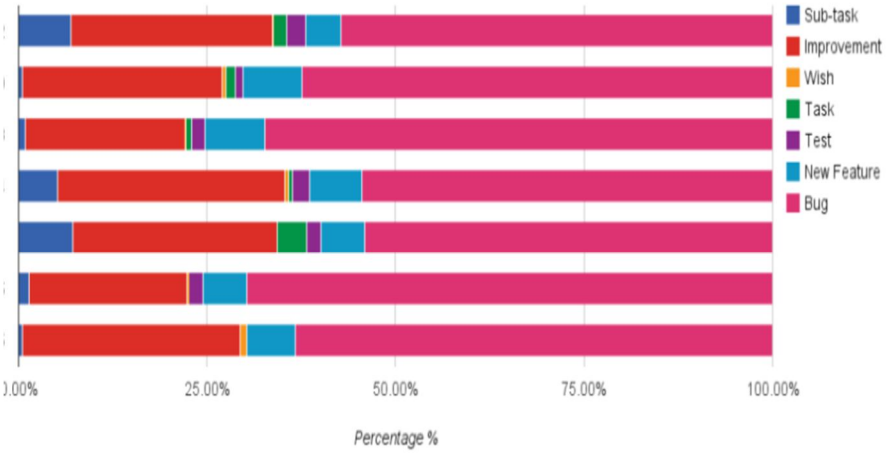
Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
14	2	1	0.08
9	2	1	1.0
12	2	1	3.3
2	2	1	7.4
8	4	1	14.2
6	2	2	14.6
1	127	1103	39.1
0	230	965	96.7
7	268	2227	97.8
3	93	386	98.2
10	2	2	100.3
5	50	114	133.8
11	2	1	379.9

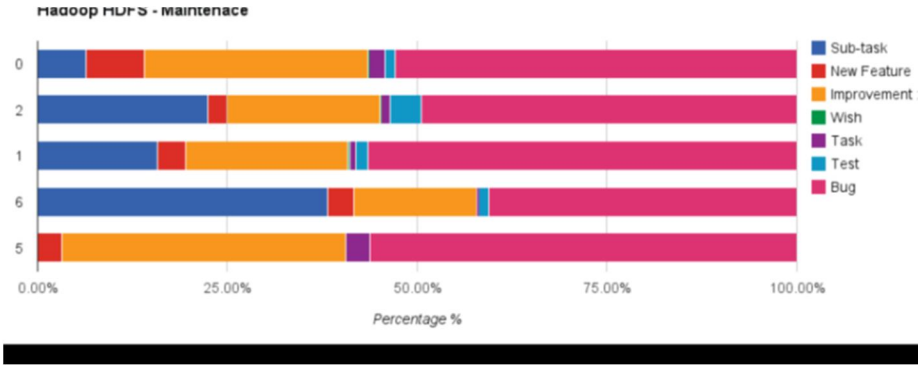
Table IX: Lucene - Core communities' statistics

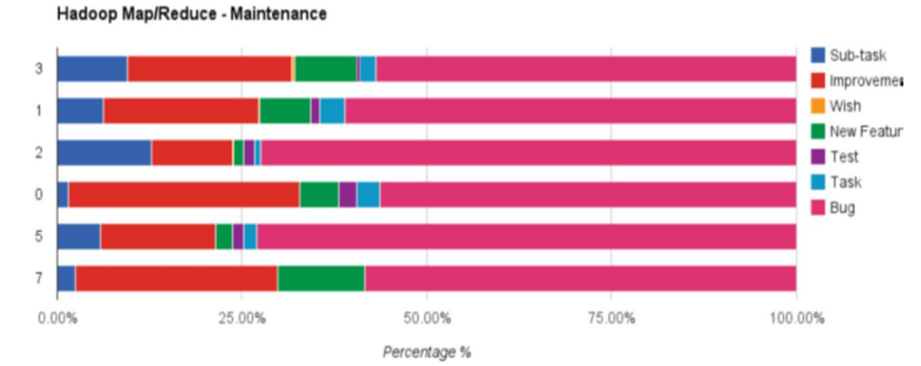
Community Id	Community Size	# of Fixed Issues	AVG Fixing Time [Days]
7	95	2	3.0
14	2	90	28.5
17	215	609	112.8
8	8	8	147.3
13	274	378	208.4
11	87	10	251.5
10	94	9	300.5
20	72	4	323.4
19	198	8	436.8
16	71	47	549.6

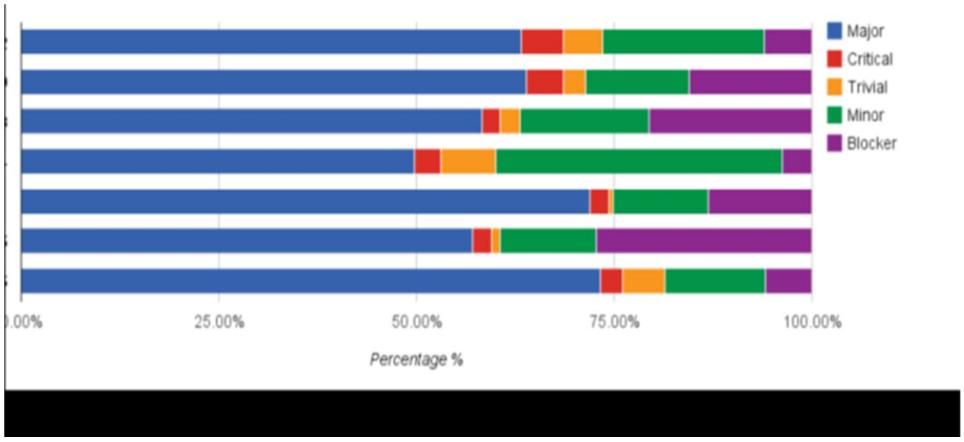
V. THREATS TO VALIDITY

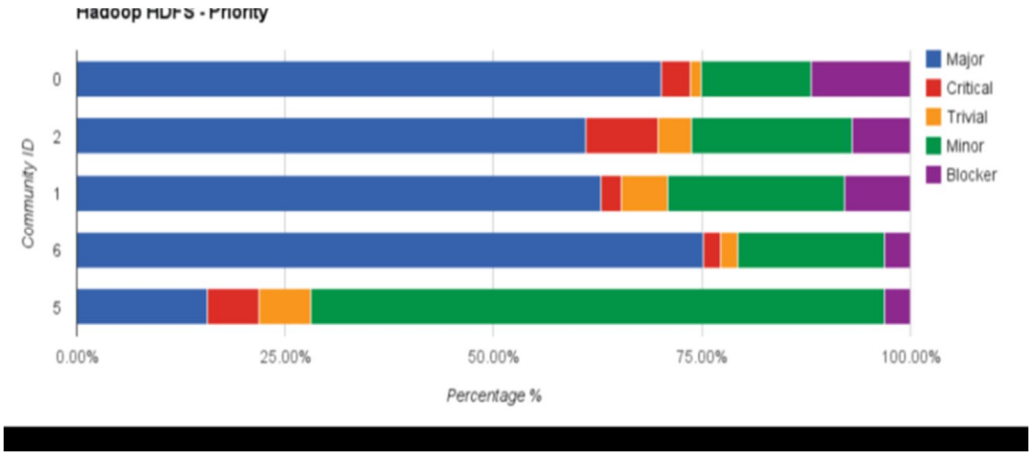
We have a tendency to currently argue on the threats for the validity of our study, following same pointers for empirical studies [26]. Manufacture authority warnings care the relation among theory and observation. For the calculation of the problem resolution time, we've got not taken into consideration the quality of a given software system or the quality of a selected sub-system of a software system. Our model may be a start line, however to get a lot of precise it'd be necessary to insert within the model a quality issue. Developers performing on the core part of the e-commerce system would wish longer to resolve a given issue associated with a region of the system that performs payment-operations, than developers concerned within the same project, however performing on a web-page that visualizes Associate in Nursing item within the basket. Threats to the centralized authority concern our selection of subject systems, tools, and analysis methodology. Regarding the system studied during this work, we have a tendency to thought-about solely seven systems hosted in JIRA. In the study, we have a tendency to create the developer network graph within which every Node represents a developer United Nations agency commented/posted on a problem, and every Edge represents a developer United Nations agency commented on there could be different ways in which to outline developer networks, considering for instance temporal neck of the woods, or link among developers redaction constant sections of code. Graphs engineered considering these factors would result in getting totally different results. Threats to external validity square measure associated with the generalization of our conclusions. Our results don't seem to be meant to be representative of all comes hosted within the repository and that we have analyzed solely the JIRA issue-tracking system.











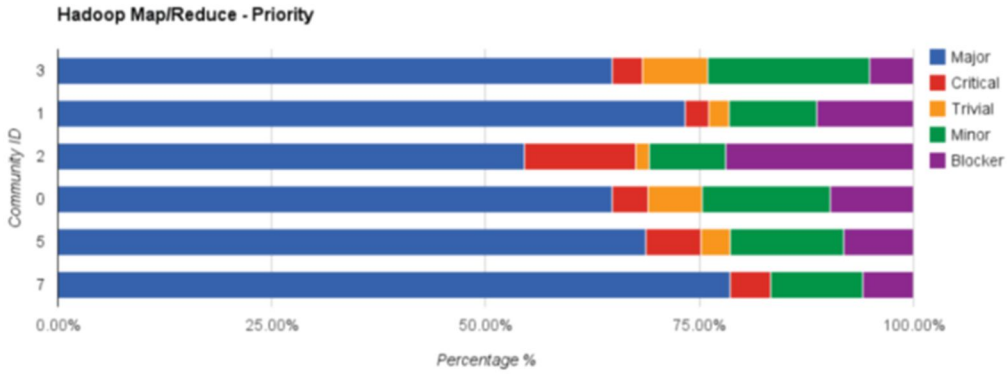
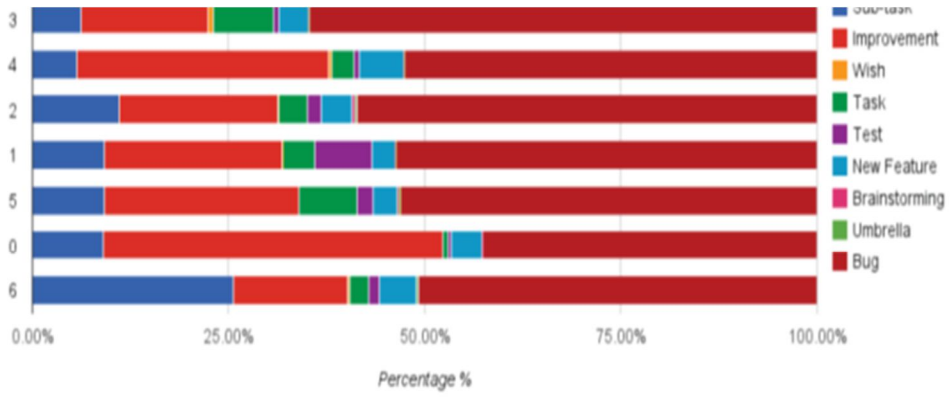
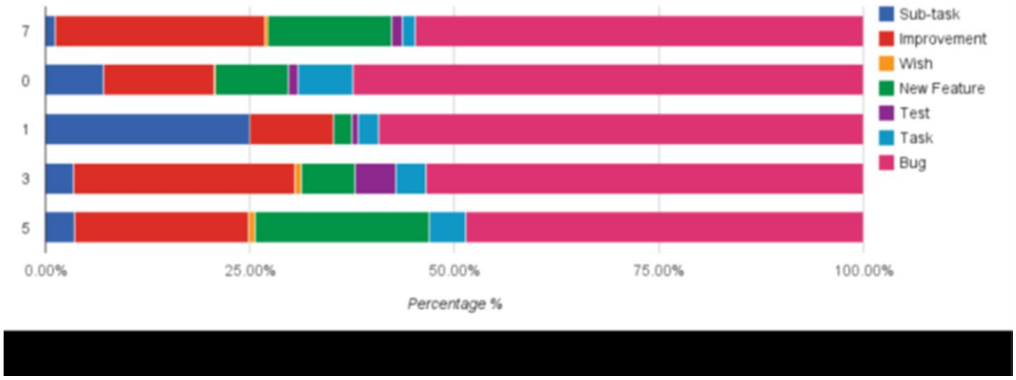


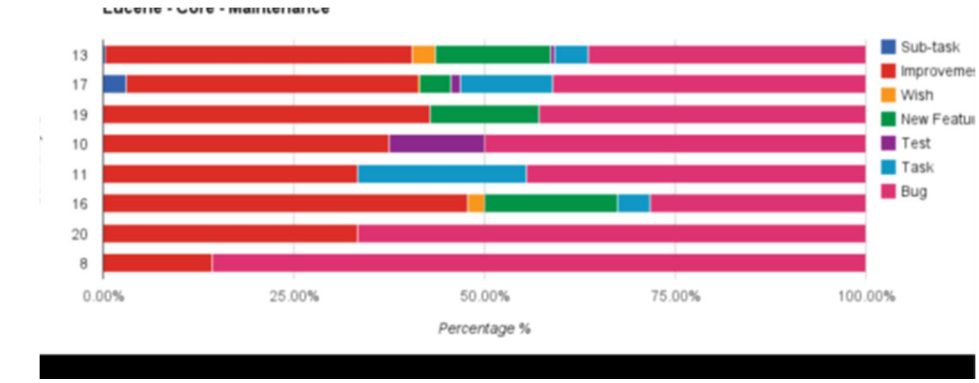
Fig. 3: examples of communities' distributions of issue's maintenance types and priorities

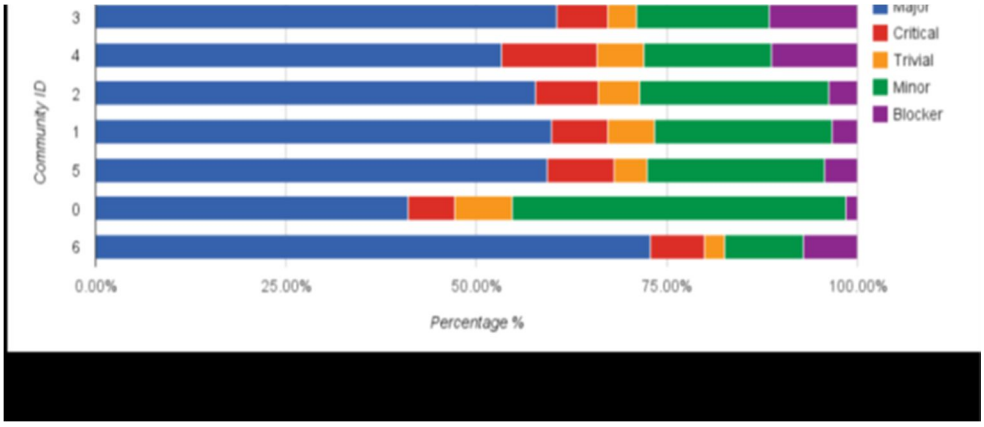
VI. CONCLUSION AND FUTURE WORKS

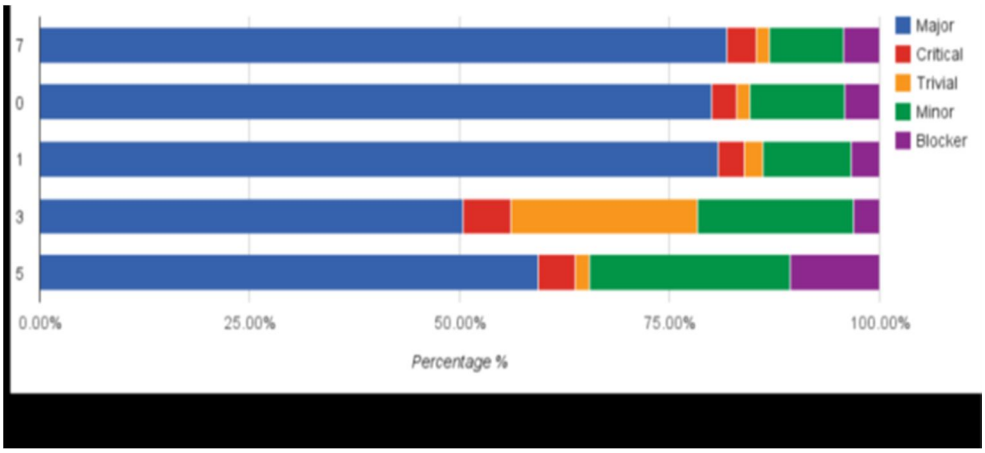
During this study, we have a tendency to analyzed the developer networks of seven open supply comes hosted in JIRA. We have a tendency to found the developers communities presence for all the results analyzed. These results agree with different studies regarding the structure of open source comes. We have a tendency to any investigated however across the communities, productivity is distributed. To live the productivity, we have a tendency to thought-about factors like the community size, the quantity of fastened problems, the distribution of fastened issues' maintenance kind and priority, and also the average issue fixing time. As a primary result, we have a tendency to found the Pareto's law presence (20% of developers doing eightieth of the work), there are unit many developers that posted and investigated the bulk of problems. The Pareto's law presence desires any investigation, one could expect this is often because of the character of the JIRA issue following system and also the structure of the ASCII text file community, and the way we have a tendency to engineered the developer's operating network. For instance, there could also be a gaggle of core developers dedicated to news problems. We have a tendency to find the typical community issue fixing time and that we found it varies across the communities. We have a tendency to show that the independence of the typical issue resolution time from the opposite issue thought-about, like the community size and also the reasonably problems maintenance and priority. Several different factors which will impact the typical community issue fixing time, for instance, the computer code element concerned within the issue resolution or the portion of code concerned. To raised perceive community productivity, we'll analyze these factors in future studies. This study may be a start line to raise perceive however teams of developers perform once operating along. The difficulty resolution time may be a helpful metric that represents the productivity of an explicit community. A promising study might be associated with the optimum range of developers concerned in a very project. A motivating future experiment might be to analyze if diseconomies of scales occur in development activity. Considering this "dilemma" as Associate in Nursing operational analysis drawback may lead to promising results.











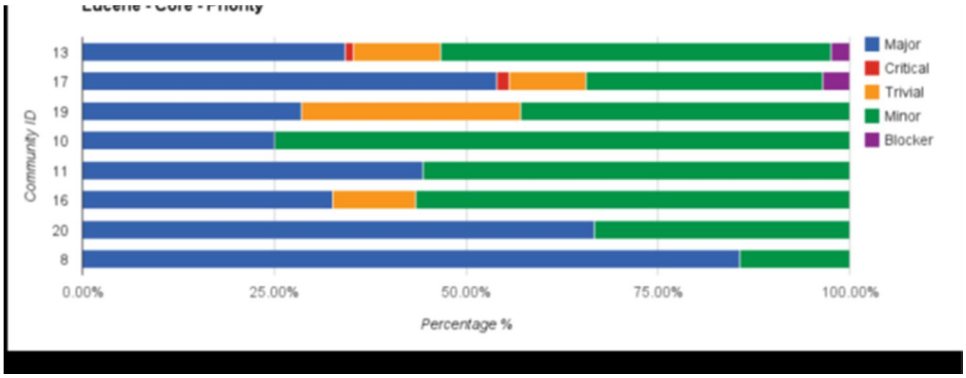
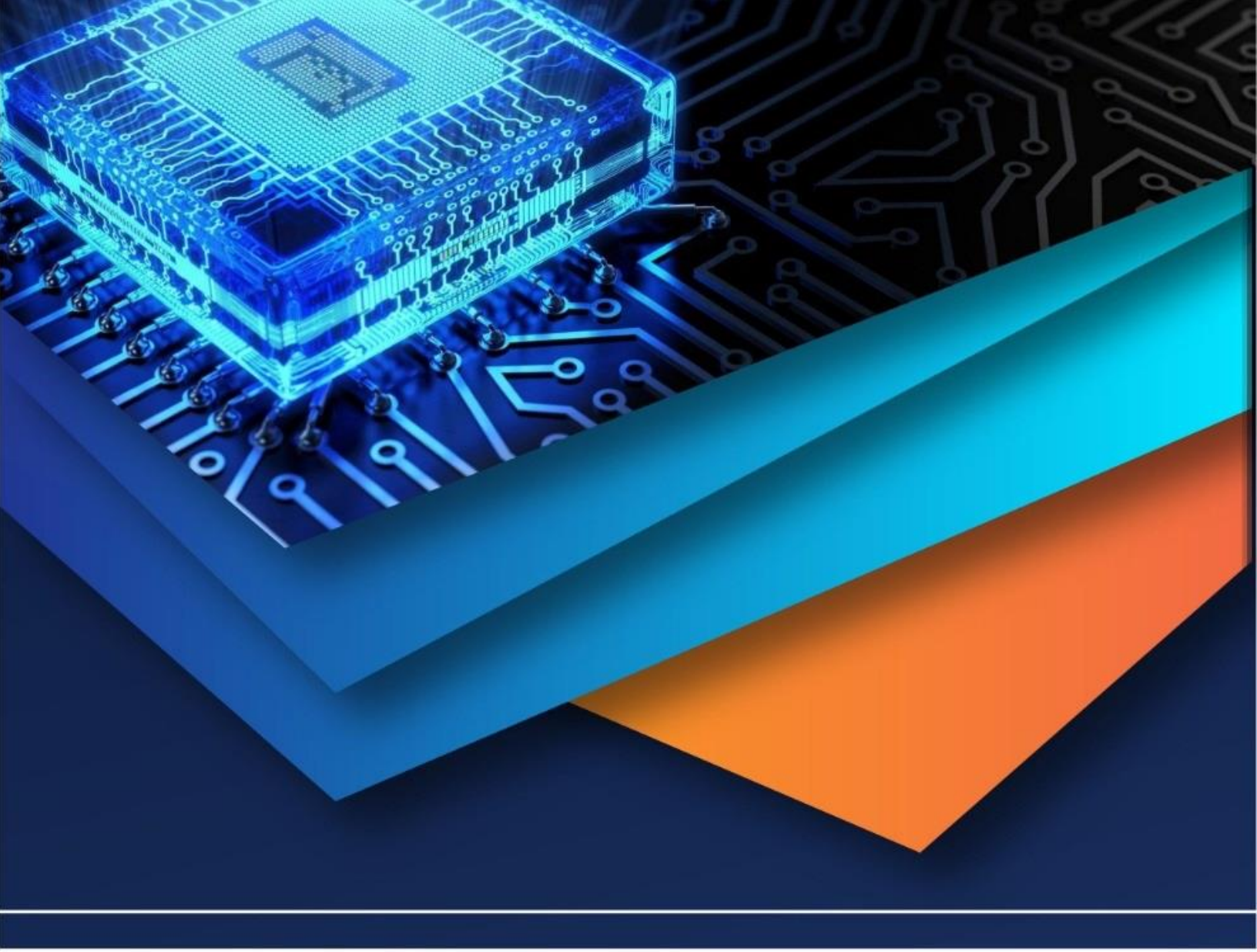


Fig. 4: examples of communities' distributions of issue's maintenance types and priorities

REFERENCES

[1] A. Capiluppi and M. Michlmayr. From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects. In Open Source Development, Adoption and Innovation, pages 31–44. Springer, 2007.
[2] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, and J. Howison. Self-organization of teams for free/libre open source software development. Information and software technology, 49(6):564–575, 2007.
[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10):P10008, 2008.
[4] G. Destefanis, S. Counsell, G. Concas, and R. Tonelli. Software metrics in agile software: An empirical study. In Agile Processes in Software Engineering and Extreme Programming, pages 157–170. Springer, 2014.
[5] K. Ehrlich and K. Chang. Leveraging expertise in global software teams: Going outside boundaries. In Global Software Engineering, 2006. ICGSE'06. International Conference on, pages 149–158. IEEE, 2006.
[6] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. arXiv preprint arXiv:0812.1770, 2008.
[7] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. ICWSM, 8:361–362, 2009.
[8] A. Hars and S. Ou. Working for free? Motivations of participating in open source projects. In System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on, pages 9–pp. IEEE, 2001.
[9] M. Ortu, G. Destefanis, M. Kassab, S. Counsell, M. Marchesi, and R. Tonelli. Would you mind fixing this issue? qn empirical analysis of politeness and attractiveness in software developed using agile boards. In Agile Processes, in Software Engineering, and Extreme Programming, pages 129–140. Springer, 2015.
[10] I. Steinmacher, T. U. Conte, M. Gerosa, and D. Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. In Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing, pages 1–13, 2015.
[11] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. Nature, 393(6684):440–442, 1998.
[12] R. K. Yin. Case study research design and methods third edition. Applied social research methods series, 5, 2003.
[13] N. Bezroukov. A second look at the cathedral and the bazaar. First Monday, 4(12), 1999.
[14] K. Crowston and J. Howison. The social structure of free and open source software development (originally published in volume 10, number 2, February 2005). First Monday, 2005.
[15] A. Meneely, L. Williams, W. Snipes, and J. Osborne. Predicting failures with developer networks and social network analysis. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pages 13–23. ACM, 2008.
[16] K. Y. Sharif, M. English, N. Ali, C. Exton, J. Collins, and J. Buckley. An empirically-based characterization and quantification of information seeking through mailing lists during open source developers' software evolution. Information and Software Technology, 57:77–94, 2015.
[17] A. Bonaccorsi and C. Rossi. Why open source software can succeed. Research policy, 32(7):1243–1258, 2003.
[18] S. Krishnamurthy. Cave or community?: An empirical examination of 100 mature open source projects. First Monday, 2002.
[19] I. Steinmacher, M. A. G. Silva, M. A. Gerosa, and D. F. Redmiles. A systematic literature review on the barriers faced by newcomers to open source software projects. Information and Software Technology, 59:67–85, 2015.
[20] M. Zhou and A. Mockus. What make long term contributors: Willingness and opportunity in oss community. In Proceedings of the 2012 International Conference on Software Engineering, pages 518– 528. IEEE Press, 2012.10
[21] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu. Chapels in the bazaar? Latent social structure in OSS. In 16th ACM SigSoft International Symposium on the Foundations of Software Engineering, Atlanta, GA, 2008.
[22] E. S. Raymond. The cathedral and the bazaar, 2000. Available from World Wide Web: http://www. catb. org/~ esr/writings/cathedral-bazaar, 2004.
[23] S. K. Shah. Understanding the nature of participation & coordination in open and gated source software development communities. In Academy of Management Proceedings, volume 2004, pages B1–B5. Academy of Management, 2004.
[24] R. Koch. The 80/20 principle: the secret to achieving more with less. Crown Business, 2011.
[25] G. Madey, V. Freeh, and R. Tynan. The open source software development phenomenon: An analysis based on social network theory. AMCIS 2002 Proceedings, page 247, 2002.
[26] P. Wagstrom, J. Herbsleb, and K. Carley. A social network approach to free/open source software simulation. In Proceedings First International Conference on Open Source Systems, pages 16–23, 2005.
[27] J. Xu, Y. Gao, S. Christley, and G. Madey. A topological analysis of the open source software development community. In System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, pages 198a–198a. IEEE, 2005.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY