CONTINUITY OF DEVELOPER RANKING AND GROWTH CHANGE PREDICTION

Anil Kumar

M.Tech Scholar, Galgotias University, Greater Noida, UP, India

Abstract

Enormous amounts of raw content and information exist universally and large databases contain unordered, ungraded and unranked data. Ranking is most famous, ubiquitous and comprehensive techniques to build hierarchy of unordered group of items by calculating rank of every item based on one or several multiple attributes values. This technique allows analyzing and evaluating product performance with other products. Widespread usage of ranking technique represents relationship among several groups of well known items.

In this analysis paper, We retrieve promising information from Git repository and demonstrate important fact of developers working individually or in a group and rank the developers based on core activities and contribution on several projects, bug resolving processes, source code commit and expertise in multiple languages by mining Git (a version control repository system). We discover key developers, influential software practitioner, projects and programming languages. We found developers grow over time in diverse areas. Our result shows developer ranking can assists project manager to take better quality decision to assign projects and gain help from expertise developer to support newly joined team members in the organization. Ranked developer can assists business to improve source code quality, timely delivery of projects, lower maintenance cost and better customer satisfaction.

Keyword - *software practitioners, ranking, developers, Language-Language, commits*

--***--

1. INTRODUCTION

Ranking has important aspects to assists users to explore popular and high quality content over low graded, unclassified, unorganized and unordered content which guide to take better quality decisions. It allows us to remove impurities and incorrect content which could provide fashionable, manageable and interesting content. Ranking is measurement of each subject and objects attribute single-attributes and multiple- attributes. For example personal favorite movies disks and list of songs, magazines, newspaper and sport items etc. Several ranking technique exists globally but visualization, comparison, analysis and performance of this techniques has to be performed to get insight into it and its results. Software developers works on several projects, resolves bugs, commits and write program code in multiple programming languages in different team in their work life cycle. This paper uses term developer or author or software engineer, software practitioner interchangeably with developers. Discovering and Distinguishing dedicate, committed, experienced, popular and expertise developer for particular field in the software project is critical issue for senior managers in the big organization. Managers always have challenge for selecting expert members from large software team. If any problem occurs in specific component or source code in big projects, then expert developers of similar projects of same team or other team is assigned to resolve the bug to increase the efficiency of the work and improve coding standards. Identifying individual experts in large team of software projects is challenging task since large project is spread over continent. Individual expertise in several project activities and leadership behavior is also mandatory to help team with different mindset in large groups. Highest number of experts in the team results in lower risk and increase performance and productivity whereas software team with less experience will end with loss of cost, timely delivery and lower quality source code and decreased productivity. They can ask a list of related questions to all team member to select the key developer could be possible but if there could be a tool or some sort of method to indentify key developer would be a better option. The tool that we could produce is to rank developers based on functional areas, key work and activities, expertise on projects, project type, kind of programming languages worked on, efficiency in identify bug and resolving bug, motivation, commitment, personal and professional behavior, and characteristics of over all work and activities in past projects in overall career. Next section described related work. Section 3 discusses methodology. Section 4 is about result observation and analysis on particular project and programming language. Section 5 discusses about overall results and section 6 is about conclusion.

2. RELATED WORK

Analyzing software projects has been deepen and resulted in improved metrics but little analysis successfully proceeded with identifying key developers and ranking those developers in diversity of areas. Some of few works have done in Jifeng Xuan et al. [1]. Their aim is to rank contribution of the developers by addressing the problem of developer prioritization in bug repositories. Explored two different methods namely model the developer prioritization and assisting predictive task with new model. They investigated three different problems including the developer ranking based on products, evolution over time and tolerance of noisy comments and also considered strengthen developer ranking to improve three predictive task i.e. bug triage, severity identification and reopened bug prediction in bug repositories. Investigated the performance of the newly model and its application in bug repositories of Eclipse and Mozilla and also analyzed the developer prioritization evolution over time. The results show that the developer prioritization can assist software tasks in bug triage. PageRank [11] and Topic Sensitive Page Rank [12] techniques discover significant node in network graph. PageRank generally used by Google to discover all directly or indirectly connected node on website spread globally and show ranked pages as result set and TSPR highlight search word on searched website pages based on topic of the context. Thung Ferdian et al. [2] they investigated the network structure of social coding in GitHub. They distinguished leading developers and projects on sub network of GitHub by PageRank approach. It helped developers in performing their tasks more efficiently by understanding how developers and projects are actually associated to each others. Found out the relationship among projects, relationships among developers and most influential projects and developers. They analyzed that projects networks in GitHub are more interconnected than human built network. Project networks only need one common developer to establish a relation between projects. In the developer-developer relationship, enables more collaborations among developers in social coding. Finding influential projects and developer on the basis of page rank gives more developer with many projects. Their results show that distribution of project-project network graph generally follows power law while developer-developer does not. Our language-language network is also extended version of their developer network algorithm. Igor Steinmacher et al. [4] describe study on newly joined authors joining software projects process since they can be key and probable contributor to the software projects growth. Their study discovers difficulties faced by newly joined authors and tried to verify usefulness of the asked queries. Their results show that retention rate is below 18 percent in mailing list and 13 percent in the issue manager and pointed out some expected cause for leaving the project. Overall analysis presents, new comer is not much interested to join past project due to inconvenience caused by improper response and clarification on particular doubt. They understood how software practitioner collaborate on software project and how new-comers behavior changes in project team because this aids management to take further decision on retention activity. Andrew Dittrich et al. [3] described a method to model a network as software projects from control version repository. They demonstrated a technique to identify the key developers and subject matter expert who work together in group and how closely meet the developers on projects. Their further investigation done to find which groups of authors work together and how closely join developers on a project. They used three different type of algorithm that gave the best results were greedy method, the modularity maximization method and spin glass method and performed on three specific open source projects namely Subversion, Audacity and Super TuxKart. Their method assumes that modifications made by each author are common and relevant to the file being modified but this is not always true in particular cases. According to their network analysis technique, this can predict the core developers for specific projects and measure probability of developer work together on the same area of code. Chen-Te Li and Show-De Lin [5] proposed several measures for example central-node based, similar node based and diverse relation based and tensor based mechanism to identify central graph nodes in heterogeneous social network and further extend it to perform role based clustering technique to identify node which could result in network of similar roles. They collect all types of relationship exists between direct and indirect nodes and captured all relational-path between nodes and adjacent nodes. Jitesh Shetty et al. [6] discover significant nodes based on graph entropy as event-basis applied on huge set of email communicated data set of Enron. They used label based graph to point lead nodes making gap of individual relation. Wasserman et al. [7] higher central node score recognize significant member with the substantial hierarchy in the network. These members would be assumed to have a significant role in simulated and regular behavior. This also applies to different kinds of network. Linton C. Freeman [8] defines set of measure of central nodes on basis of betweenness of group of neighbored nodes. Their measure is point and graph centrality based which defines degree of central nodes falling point on shortest path between each other. Kazuya Okamoto et al. [9] combine past closeness centrality measures and discover a new algorithm to rank top vertices based on highest measure value of closeness centrality. Their algorithm performs faster than expected when applied on all vertices to calculate closeness centralities. Douglas R. White and Stephen P. Borgatti [10] reviewed work of Freeman’s centrality measure of betweenness of undirected graph and purposed directed graph for the same. They included point centrality, distinct maximum central node graph for direct graph, incoming and outgoing arc and arcs for maximum central structure. They also considered relative betweenness and individual points. This extended work on directed graph improves the find in the large network with direct connected-relation.

3. METHODOLOGY

This section we demonstrate our methods for building a sample software entity network from GitHub repositories. We present our new algorithm to analyze the network.

3.1 Data Set Collection and Processing

Enormous amount of duplicate and distinct data is available world-wide in several formats. Our work is based on Git distributed and scalable version control repositories of GitHub. Git (http://git-scm.com/) is the most recent data freely available globally which contains hug data in complex and structured format. GitHub is source-code management site which hosts around millions of software repositories and managed by millions of regular and registered software practitioners. As part of our work, investigated 108718 random projects, users (498574) out which distinct developers (83604) worked on 108616 different projects, Commits(593573), Bugs(149821), Bug Comments (534104) and 20 distinct programming languages used on multiple projects.

3.2 Building Network Structure

We build several kinds of software entity network from GitHub huge and complex dataset - a language-language network, a developer based network and bug-bug based network, follower based network and network based on number of developers taking care of other projects. Network is built through directly or indirectly connected nodes and edges with or without weight. Some network graphs are based on weight. Weight is assigned with the number of incoming links to nodes. For example if particular node has five incoming links or nodes meaning weight of that node becomes five. Unweighted network structure graph does not contain any predefined values. Languages have at least one common software practitioner or developer. While building language-language network, we present a set of step presented in Algorithm 1. Each step will show how the language based network is being constructed. Each Steps are as follows: We initially pass language name as input to the algorithm, Get all available language as a set. For each language from the set, retrieve the list of projects that are based on language in the list, then find all available developers who works on the projects, get set of projects in which developer are directly or indirectly involved and finally list of available language is compared with input language. Below is the basis idea represented in the form of algorithm to build language-language network. Other entities network building in this paper is based on this algorithm.

3.3 Language-Language Network building

Algorithm 1

```
Input: Languages  // List of programming languages
Linked-Network ← Ø ; // Language-Language network
foreach language Li in Languages do
    Projects ← ListProjectLanguage(Li)
    foreach project Pi in Projects do
        Developers ←ListAssociatedDevelopers(Pi)
        foreach developer Di in Developers do
            ReturnedProjectList ← listProjects(Di)
            foreach project Pj in ReturnedProjectList do
                setofLanguage ← ListLanguages(Pj)
                foreach language Lj in SetofLanguage do
                    connection ← CountCommonDevelopers(Li, Lj)
                    Linked-Network ←{ Linked-Network, connection }

return Linked-Network
```

3.4 Sample Snapshot of Language-Language Structural Network

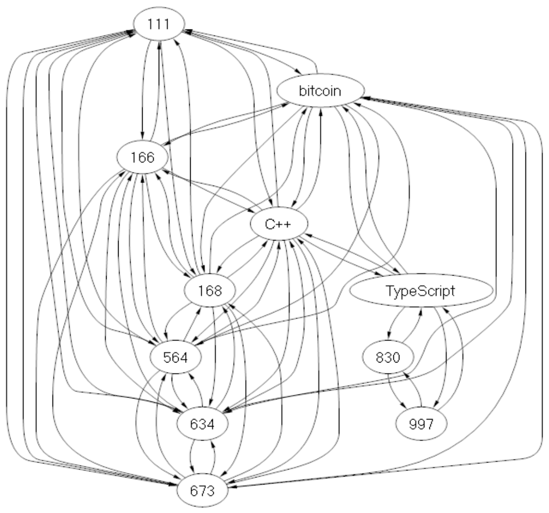

Fig 1: Demonstrates network structure of language based network.

Snapshot showing in the Figure1 contains 66 edges and 11 nodes in this language based network hierarchy. We can describe as one project (bitcoin) node having two different languages (TypeScript and C++) node. Developer nodes (830 and 997) directly associated with language (TypeScript) to (bitcoin) project node and several developers except 830 and 997 nodes are directly connected to (C++) language node and (bitcoin) project node. Relationships among languages and developers have also been shown. Since C++ and TypeScript are type of language node. Relationship between developers, project and language node have been illustrated. It shows that they all collaborate on the same projects and have same type of relationships. Bitcoin project node is central node representing language node and developer nodes structurally with the help of common relationship among developers and languages.

4. RESULT OBSERVATOIN AND ANALYSIS

We execute the newly designed algorithm on our GitHub dataset and perform result analysis. Our observation results show in terms of growth of several entities including developer, language and commit. Ranking evaluation of these entities has been performed.

4.1 Brief Data Set Survey on Language Associations

This section demonstrates interest growth, advancement and impact in each and specific software entity. We describe each entity taking specific advancement history in it. Language history for over six year shows that selections of particular programming language for critical projects have changed and some other language is taking place. Selections of object oriented language have also remained constant but decreased irrespective of user-friendly development environment. Programming languages are critical part for projects and developers selection. If the existing team does not have expertise in particular language and assigned for project then that could make big difference to team and projects.

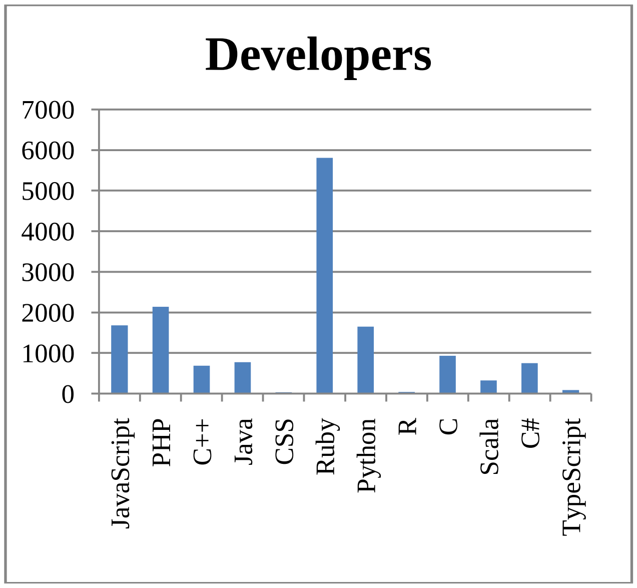

Chart 1 : Developer interest growth in the particular language

Chart 1 result shows that Ruby is favorite language for software developer. Other language for example PHP, JavaScript and Python is also competing Ruby. Object-based and oriented language like C, C++ and C# are bit below developer expectation.

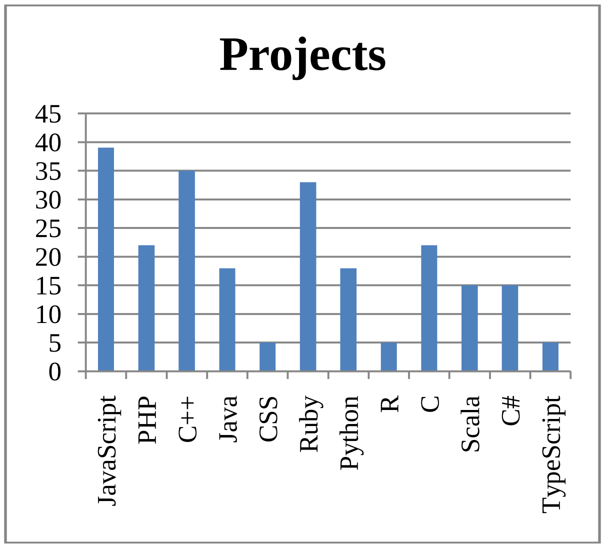

Chart 2: Listing project growth in top most languages

Chart 2 Project-wise result in listed language shows that JavaScript, C++ and Ruby are chosen mostly among other languages by project manager. In this result, JavaScript and C++ are better choice than Ruby, which is developer favorite choice in scripting view point. These selected projects are based on developer collaboration and shared sourced code basis. Only top most languages have been demonstrated here to present growth of project in each language. Most struggling programming languages are C# and R and others are CSS and TypeScript. These results are based on developer choice over common projects.

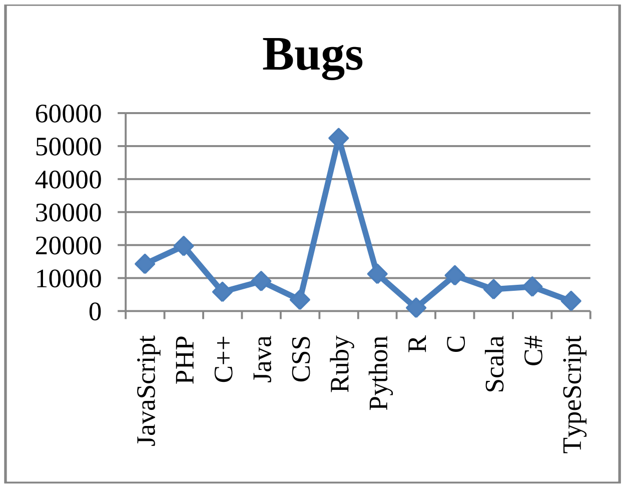

Chart 3 : Language-wise overall bug reported

Result shows that more bugs were reported in Ruby and PHP in comparison with other ideal languages. Object-oriented languages for example C++, C# and Java performs better shown in chart 3. Since object-oriented language have always been less bug producer, developers can have faith on it while writing software programs on critical and important software projects. Around 5811 developers on unique projects reported 52426 bugs on Ruby language which is the highest among all. Again all these results analysis are based on common relationship between developer, projects and language.

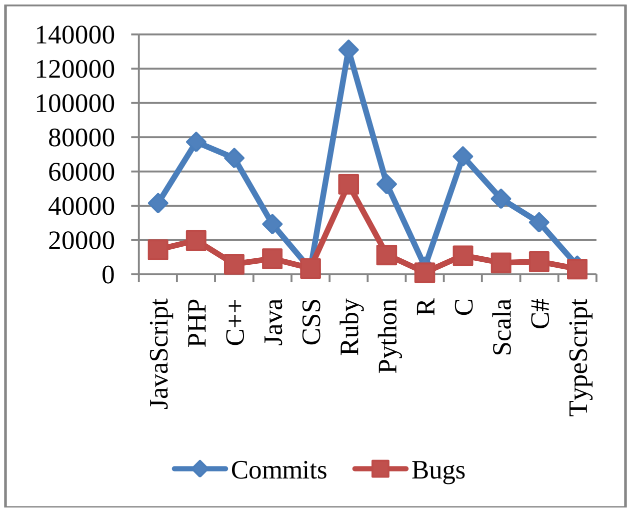

Chart 4 : Commits and Bug report comparison

Chart 4 present more commit and less bugs reported. As we observe PHP, C++ and C have got more commits but less bug whereas Ruby commit more and generates more bugs. Software practitioner and software development management team can take decision based on commits and bugs reported by users and developers in specific language. Choosing particular language is critical decision when fresh projects team start working.

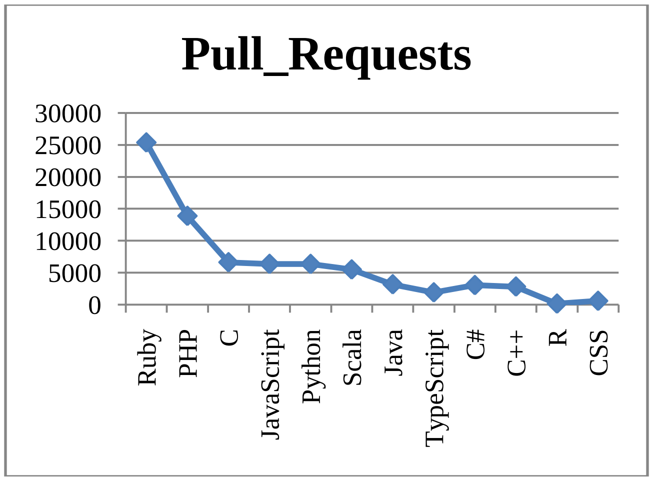

Chart 5 : Language-wise source code pull requests

Pull request is notification to project maintainer about modification in project fork. It is about discussion about potential modification on source code and code review on changed code set. It is sum of review comparison, Bugs reported and commit comments. Ruby has the highest pull request whereas CSS has the lowest among other scripting languages as shown in chart 5.

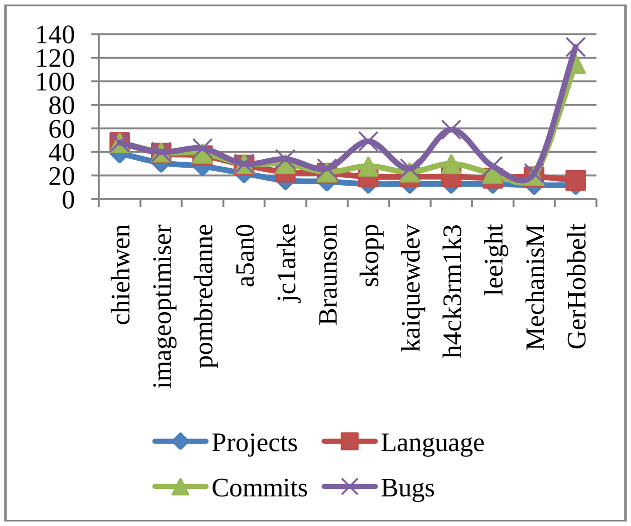

Chart 6 : Developer contribution distribution

Chart 6 demonstrates developer contribution in multiple project activities. Top developers contribute and share their expertise and assisting other developers to improve their quality of code, timely delivery and bug less project completion. Different developers have specific qualities that help others. Some developers might be expertise some language but they can fix more bugs on several languages and some might have worked on some projects but have less bug resolving activities.

Developer (MechanisM) has committed more source code and reported more bugs also. Chiehwen and imageoptimize and pomebredanne worked on the highest number of projects and languages. However, all developers are the best performer and contributor on multiple projects and programming languages.

4.2 Year-Wise Statistics

Year-wise statistics shows growth of developer and languages improves over time.

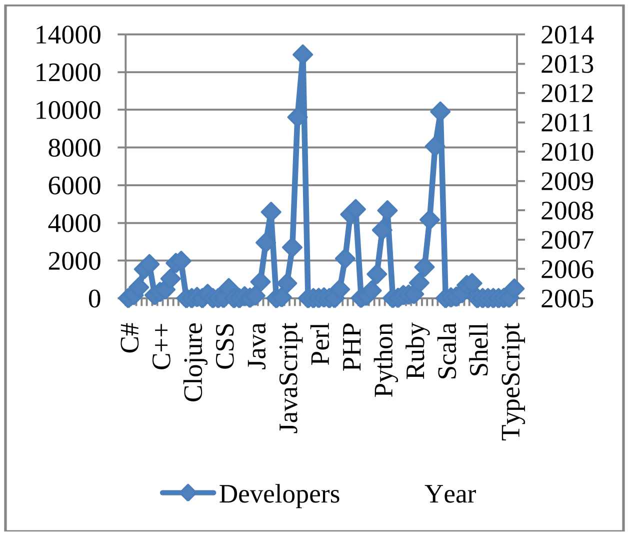

Chart 7: Developer interest in languages

Developer interest in various programming languages has changed continuously year-to-year. Several new language came to picture but not able to survive for more period of time.

Above result shown in chart 7 predict that JavaScript and Ruby have been in great demand of developers. Software practitioner contributes to multiple projects based on work experience on particular programming language. Project manager assigns projects to developer and language specific team to increase productivity of the team and successful completion of the software projects.

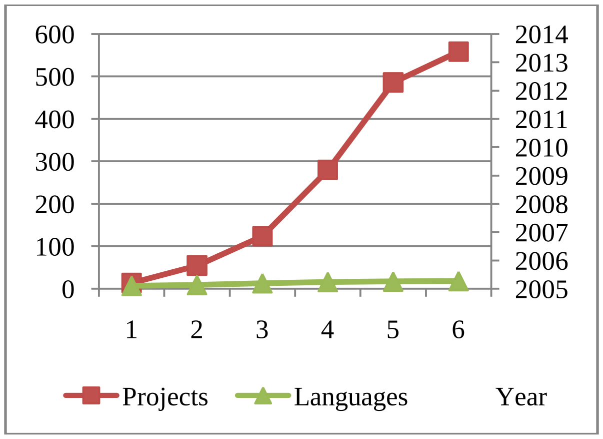

Chart 8 : Growth of project and language

Tremendous growth of projects changes over year-to-year whereas language remains constant. Drastic increase between years 2010 - 2012 in projects growth shows that multiple developers joining on the projects and the organization has change over time.

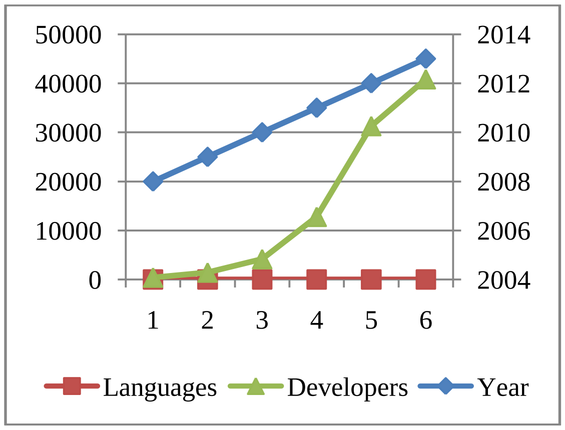

Chart 9 : Continuous growth of developers over year

Continuous growth of developers increased in few years whereas language remains constant. Drastic increase in developer's growth shows that joining on multiple projects and the team has increases over time. A developer switches to multiple languages when projects get completed and new project is assigned to work on. Languages also get changed when some specific component of the same project is to be developed on the same language. Developers have to choose other language irrespective of expertise, when new projects are assigned to their team and their role gets changed.

4.3 Ranking Entities

Table 1: Developer ranking based on contribution to multiple projects

Project-wise developer ranking			
Developer	Rank	Dense Rank	Projects
Philippe Ombredanne	1	15	28
imageoptimiser	2	16	31
Eugene MechanisM	3	10	12
Braunson	3	12	15
John Clarke	3	13	16

Table 2: Developer ranking based on expertise in several languages

Language based developer ranking			
Developer	Rank	Dense Rank	Language
Philippe Ombredanne	1	9	9
Imageoptimiser	2	8	8
Alex Schoof	3	7	7

John Clarke	3	7	7
Braunson	3	7	7

Table 3: Source code committed by developer

Commit-wise developer ranking			
Developer	Rank	Dense Rank	Commits
Fabien Potencier	1	162	7903
Tenderlove	2	161	3158
Asparagui	3	160	2575
Bnoordhuis	4	159	2400
Kevinsawicki	5	158	1992

Table 4: Developer activities on bug reports

Bug-wise developer ranking			
Developer	Rank	Dense Rank	Bugs
Fabien Potencier	1	119	2863
Rafael franca	2	118	2309
Ben Noordhuis	3	117	1770
Steve Klabnik	4	116	1619
Parker Moore	5	115	1450

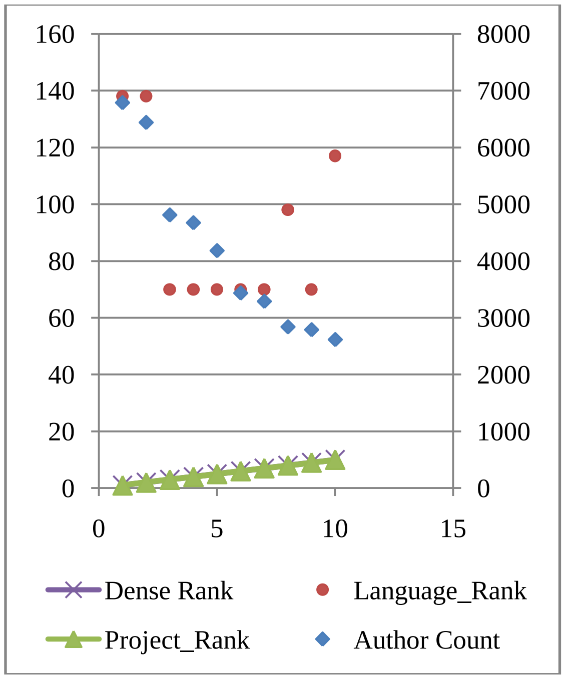

Fig 2 : Project Rank vs. Language Rank

5. OVERALL RESULT ANLAYSIS

Rank Analysis: Developer ranking is technique to assign highest priority to every software practitioner in GitHub and prioritize his task and contribution of individual and team to aid specific software jobs. Ranking has been done based on Microsoft ranking algorithm. Analyzing table 1 and table 2, we find that Philippe Ombredanne, Imageoptimiser and John Clarke are common in Project and Language based ranking among top 5. This shows that they have common activities and deep relationship between them. In same way table 3 and table 4 have developer Fabien Potencier in common with bug resolving and source code commits. Philippe contributed on twenty eight different projects and nine programming language but Imageoptimizer worked on thirty one software projects and eight language which is one less than Philippe.

We discover evolution of software practitioner ranking for longer period of time. We demonstrate that developer ranking is most effective way to improve work and increase productivity in team coding based on identification of key developers. Developer ranking can increase more information to decision making system. The results are divided base on different factor namely priority of developer over number of projects, developers interested on selected programming languages, number of bugs reported and further activities on the those works. Developer involvement in source code commits has been taken into account. We extract several kind of information based on common activities on projects works, languages participated and expertise in the language, bug reports on the multiple projects and committed source codes on the same projects or same kind of projects. Fabien Potencier contriute to source commit in several projects. He committed around 7903 times and Kevinsawicki committed only 1992 commits which is in among top 5 ranks. Fabien Potencier and Rafel Franca reported 2863 and 2309 bug reports which is maximum among others developers.

Our result demonstrates that a group of developers have high contribution to several projects. Some have contributed to specific language and have performed well on that language. Few developers who have high contribution to commit and bug reports and resolving bug have not in top list of developer and language list. They have better contribution to their own interest group. For example Philippe Ombredanne, Imageoptimiser, Braunson and John Clarke are top rankers in language and projects based work. They have contributed to multiple projects and languages but not have much contribution in commits and bug reports activities.

Our results can predict that each developer can have expertise in multiple languages and contributed to several distinct projects but not in the bug reports and source code commits. Developers having good command over bug reports can get testing bug reports role in future projects. Developers having expertise in multiple languages can assure that they can be a part of decision making team when new projects is being designed and assigned to a group of developers in the organization.

6. CONCLUSIONS

This paper described developers, projects and language growth in specific period of time and long periods. We extract Git version control repository system to model our network and performed analysis. We extracted 108718 random projects and 498574 regular users and 83604 unique developers who were common in projects work and language specific work. Our overall result is based on common developer activities on several entities.

REFERNCES

[1] Jifeng Xuan, He Jiang, Zhilei Ren, and Weiqin Zou. "Developer prioritization in bug repositories" , ICSE, page 25-35. IEEE, (2012)

[2] Thung, Ferdian; LO, David; and JIANG, Lingxiao, "Network Structure of Social Coding in GitHub" (2013). *Research Collection School of Information Systems (Open Access).*

[3] Andrew Dittrich, Mehmet Hadi Gunes, Sergiu Dascalu, "Network Analysis of Software Repositories: Identifying Subject Matter Experts", Complex Networks Studies in Computational Intelligence Volume 424, 2013, pp 187-198

[4] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, Marco Aurélio Gerosa , "Why do newcomers abandon open source software projects?" ,Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on IEEE, 2013, page. 25-32

[5] Cheng-Te Li and Shou-De Lin, "Centrality Analysis, Role-based Clustering, and Egocentric Abstraction for Hetrogeneous Social Networks" 2012 International Conference on Social Computing ((PASSAT, SocialCom), IEEE, Sept 2012

[6] J. Shetty, and J. Adibi. 2004. Discovering Important Nodes through Graph Entropy: The Case of Enron Email Database. In Proceedings of ACM Workshop on Link Discovery (LinkKDD'05), 74–81.

[7] Wasserman, S. & Faust, K. (1994), "Social Network Analysis: Methods and Applications", Cambridge university Press.

[8] Freeman, Linton C., "set of measures of centrality based on betweenness." Sociometry Vol. 40, No.1, 1970, 35-41.

[9] Kazuya Okamoto, Wei Chen, Xiang-Yang Li, "Ranking of Closeness Centrality for Large-Scale Social Networks", Second Annual International Workshop, FAW 2008, Changsha, China, June 19-21, 2008, Proceeedings, pp 186-195

[10] Douglas R. White, Stephen P. Borgatti, "Betweenness centrality measures for directed graphs", Published by Elsevier B.V., Volume 16, Issue 4, October 1994, Pages 335–346

[11] Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999) The PageRank Citation Ranking: Bringing Order to the Web. Technical Report.Stanford InfoLab.

[12] Haveliwala, "Topic-Sensitive Pagerank", Proceedings of the 11th international conference on World Wide Web (2002): 517-526

[13] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E. Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi, "A Large-Scale Empirical Study of Just-in-Time Quality Assurance" IEEE Transactions On Software Engineering, Vol. 39, No. 6, June 2013

[14] Zimmermann, T. and Nagappan, N., "Predicting Defects using Network Analysis on Dependency Graphs," in 29th International Conference on Software Engineering, 2007

[15] Kim Herzig, Sascha Just, Andreas Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction", Proceeding ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, Pages 392-401

[16] Shivkumar Shivaji, E. James Whitehead, Jr., Ram Akella, Sunghun Kim, "Reducing Features to Improve Code Change Based Bug Prediction", Journal IEEE Transactions on Software Engineering archive Volume 39 Issue 4, April 2013, Pages 552-569

[17] Dongsun Kim, Yida Tao, Sunghun Kim, Andreas Zeller, "Where Should We Fix This Bug? A Two-Phase Recommendation Model", IEEE Transaction on Software Engineering, VOL. 39, NO. 11, NOVEMBER 2013

[18] Moser, R., Abrahamsson, P., Pedrycz, W., Sillitti, A., Succi,G., "A case study on the impact of refactoring on quality and productivity in an agile team", In Proc. of the 2nd IFIP Central and East European Conference on Software Engineering Techniques CEE-SET 2007, Poznan, Poland (2007).

[19] Huzefa Kagdi, Michael L. Collard and Jonathan I. Maletic1, A survey and taxonomy of approaches for mining software repositories in the context of software evolution, Journal of Software maintenance and evolution : Research and Practice J. Softw. Maint. Evol.: Res. Pract. 2007; 19:77–131

[20] Xiao Yuan Xie and Tsong Yueh Chen and BaoWen Xu "Isolating Suspiciousness from Spectrum-Based Fault Localization Techniques", In Proceedings of the 2010 International Conference on Quality Software (QSIC), 2010, page 385-392, IEEE

[21] Wong, W.E. and Shi, Y. and Qi, Y. and Golden, R. "Using an RBF neural network to Locate Program Bugs", In Proceedings of the 19th International Symposium on Software Reliability Engineering, 2008, page 27-36, 2008, IEEE/ACM.

[22] Santelices, R. and Jones, J.A. and Yu, Y. and Harrold, M.J. "Lightweight fault-localization using multiple coverage types ", In Proceedings of the 31st International Conference on Software Engineering, 2009, page 56-66, 2009, IEEE.

[23] Xutao Li, Ng, M.K., Yunming Ye, " MultiComm: finding community structure in multi-dimensional networks", Knowledge and Data Engineering, IEEE Transactions on Volume:26,Issue: 4, April 2014, pp. 929 - 941, IEEE

[24] Huzefa Kagdi, Michael L. Collard and Jonathan I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution", Journal of Software maintenance and evolution : Research and Practice J. Softw. Maint. Evol.: Res. Pract. 2007; 19:77–131

[25] Anil Kumar, “Evolution of social developer network in oss: survey", International Journal of Research in Engineering and Technology,Volume: 03 Issue: 04 | Apr-2014

BIOGRAPHIE

I graduated from Gaya College Gaya, did MBA and MCA from Manipal. Currently I am M.Tech (CSE) Scholar from Galgotials Univeristy, UP, India.