Mining Developer's Contribution Across Source Code Commits in WoC Infrastructure

Nakul Sharma[1], Balasubramnium Vivekanandam[2], Eugenio Vocaturo[3]
[1] Linclon University College; [2] Linclon University College; [3] Linclon University College
Email ID [1]pdf.nakul@lincoln.edu.my,
[2]vivekanandam@lincoln.edu.my,
[3]eugenio.vocaturo@cnr.it

Abstract: Source code mining on online repositories is an emerging research field. Mining seeks to discover interesting information and patterns from the online repositories. World Of Code (WoC) platform provides an interface to query meta-data and software supply chain about the projects across different source code repositories such as BigBucket, GitLab, GitHub etc. This work proposes the possibility of querying WoC for mining essential relationships existing between developer and the modified repositories or files. The current work also provides a metric or a scale to analyze relationship existing between the commits and developer. This scale classifies developer as active, inactive or borderline. Developer hence classified can be selected for future project work.

Keywords: Developer; World Of Code (WoC); Mining;software;repositories;

Introduction

The software mining tasks need to be conducted on different repositories. In order to accomplish this, it is essential that large scale analysis is conducted on different hosting platforms. The integration of these hosting platforms is found in software infrastructures such as World of Code. The large scale analysis of repositories both at macro as well as micro level can be conducted on such infrastructure. The World of Code infrastructure presents an opportunity to study software chains and other developer related information. The Woc infrastructure provides hosts of services related to developer and project. These include providing information using maps about developer's who commit on a repositories. It also uses awk programming to find out number commits made by the same developer on multiple projects. This helps in studying the relationship between developer and the commits done across repositories [1].
The present work identifies following information related to each repository present in World of Code infrastructure.
1. Developer's information (name, email id)
2. Amount of commits undertaken
3. Name of the repository
4. Number of files affected by a commit.

5. Time stamp of the commits
6. Frequency of commits
7. Number of repositories committed to a single developer.

The developer's activity is not just restricted to a single repository or a single hosting platform. Hence, it is essential to to query a common platform which hosts integration of different sourceforge's. WoC code infrastructure combines several such source forge and source code hosting platforms thereby enabling analysis at one place. The essential information of the developer can hence help in creating metrics and measures relating to developer's core activities.

Related work

WoC infrastructure came into existence with the research conducted by Ma Y. et. al. The fundamental reason for creation of this infrastructure was to study the software evolution and to conduct software supply chain analysis. The infrastructure integrates several code hosting platforms in order to provide an exhaustive data for analysis and study [1]. Table-1 provides a summary of work undertaken on WoC project. This section discusses some of the work conducted on WoC platform since 2019.

2020

The authors T. Fry et. al. developed a data set and model for prediction. The specific focus of the dataset was to identify author identities. The authors scanned 2B billion git commits to extract author records totaling to 38 million. This exhaustive information extraction was made possible through repository mining, heuristics and ML [2].

Another work by T. Dev et. al. discusses identification of bots from within the large scale code repositories such as WoC. This work made use of bot identification algorithm and made use of association mining rules to accomplish this task. Authors made use of exploratory and empirical studies in this work [8].

The work by T. Dev et. al. discuss both discovery and its classification through an algorithm. A dataset of 461 bots was identified. This dataset could be fed to various AI/ML models for prediction and analysis [15].

2021

Walden J. et. al. have worked on providing an empirical analysis of different project activity based changepoints in OSS community. The research work mined repositories in WoC using heuristics and employed PELT algorithm for getting dataset of projects. The authors made use of changepoint detection algorithm for evaluating the system generated. [3].

Hao et.al. have analyzed java software's evolution. The author made use of heuristics and iterative selection for conducting this analysis. The metric based evaluation was also conducted [6].

2022

Ma.Y. make use of social contagion theory and statistical modeling in order to recommend certain software libraries. The authors also found some library migration statistics. The proposed system was evaluated using social contagion model [5].

Sousa et. al. developed a dataset by using WoC infrastructure. The time based open source project dataset was extracted from WoC. This accomplished by making use of heuristics and iterative based selection model [7].

Abdellatif et. al. developed a tool named bothunter. This tool uses WoC infrastructure along with certain heuristics and ML model to develop this bot. The evaluation was conducted using standard ML techniques [13].

2023

Reid et. al. makes use of WoC infrastructure in development for developing UVHistory. This is used for tracking changes across different repositories [10].

Rosa et.al. created a multi-stage algorithm for extracting docker files from WoC infrastructure. The algorithm conducts parsing to extract the necessary files. The artifacts created were docker files and evaluation used was empirical [11].

Wang C.et. al. created a ML model using mining heuristics. The proposed algorithm helped in recommending autoML downstream topics. The empirical and statistical metrics were used for evaluation [12].

Table 1. Related Work

Reference	**Focused Area of Research**	**Specific Methodology**	**Artifact Created**	**Evaluation Metrics Employed**
[2]	Identity Resolution, Git, Commit, Software Change Management	Repository mining, Heuristics	Dataset , Model for prediction	Manual, Precision, Recall
[3]	Software Evolution, migration, repository mining	WoC Mining, Heuristics, PELT Algorithm	Dataset of projects	Change point detection algorithm
[4]	Library Migration, Software Evolution	Event-based dependency	Model for determining	Repository frequency

		changed model	dependency changes and migration commits	analysis, Migration frequency analysis, migration graph
[5]	social contagion, technology adoption,SSC	Theory of social infection and statistical modeling	Algorithm for recommending library recommendation	social contagion model
[6]	library migration, MSR, recommendation system	Heuristics based, iterative selection	Algorithm for recommending software libraries	Empirical based evaluation
[7]	Software Evolution, software metric, data mining	Heuristics based, iterative selection	Temporal and Open source based Java projects dataset	CK Tool, Metrics
[8]	Code Commits, Association Rules	Bot Identification and categorizatio n according to association mining rules.	None	Empirical evaluation
[9]	software evolution, Stack exchange, mining infrastructure.	LDA, Card sorting algorithm	XAI packages collected, trend progression	Empirical and parameter based evaluation
[10]	Version History, software evolution	Not specified	UVHistory for tracking changes across repository of WoC	Research questions based on parameters.
[11]	docker, deep learning, source code mining	Parser and multi-stage algorithm for docker file extraction	Docker file	Empirical Evaluation
[12]	AutoML, software engineering,	Repository	AutoML	Empirical,

	repository mining	mining, heuristics, ML	downstream topics, recommendations on use of Auto ML	statistical based
[13]	software bots, automation, repository mining	Repository mining, heuristics, ML	ML Model	ML based evaluation techniques
[14]	Software supply chain, WoC, repository mining	scripts to scan and join partitions of WoC	Ptb2Pt maps stored as tables	None
[15]	Bot identification, repository mining	Bot identification using heuristics	Dataset of 461 bots.	Data visualization of parameters.
[16]	repository mining, author identification, Machine Learning	Mining Author ID and their duplicates in Woc	Developer specified profile generator	Feedback From Developers, Identity disambiguation
[17]	repository mining, sustainability,	interview, WoC repository mining, heuristics	vector representation of software libraries	-

Method, Experiments and Results

The proposed methodology is categorized into two sub-sections. The first is data extractor section and second is data analysis section.

Data Extractor Section

Data extractor module algorithm is presented in this section. The data extractor module is divided into following different steps:-

1. Establishing connection with the WoC server
2. Extracting the developer and repository specific information from WoC server.

In this phase, we analyzed the data extracted from individual da5 using mongodb queries.
The WoC sections have got lots of repositories and it was not possible to extract data from each one of them. Hence the only repositories present in mongoDB database were considered. In this section, data

is extracted from the Woc folder of mongodb for the meta-analysis. It contains several columns. The results were analyzed and some developer specific metrics proposed.

Data Analysis Section

In this section the dataset is subjected to feature engineering to analyze the quality of development undertaken by the developer.

Metric Definition

Proposed metric checks the performance of developer across various projects. There are different measures to check to performance of developer. A new metric is proposed in order to check efficiency and regularity of developer in making commits.

Let c be the number of commits undertaken by the developer D_i . If count of commits for each month exceeds 5 for each repository, then developer D_i is considered active. Let devCon be the total number of files, the developer commits. The categorization of developer is named as cD_i.

Metric Formulation

if NumCommits >5
cD_i="Active"
else if NumCommits ==5
cD_i="BorderLine"
else
cD_i="Inactive"

NumCommits is a column indicating the number of commits done by developer. The threshold value of 5 was chosen as arbitrary, which could vary from project to project.

Table 2. Results of applying cD_i *metric to mongodb dataset*

Total Number of mongodb entries scanned	**Number of Active developers**	**Number of Inactive Developers**	**Number of BorderLine Developers**
144	75	36	29
%	52.08	25	20.13

Table-2 shows the result of applying cD_i scale dataset of 144 entries were taken from mongodb for analysis and scale application. It is shown that out of the list of 144 entries, 75 are active developers. Number of inactive developer's were 25 and 29 were borderline developers. According to the percentage, there was 52.08 % developers active, 25% were inactive and 20% were borderline.

Table 3. Compares this work with the related work or previous research by other researchers

Parameters	**Proposed Methodology**	**Reference Number 3**	**Reference Number 7**
Methodology Proposed	repository mining, Heuristics, iterative selection	WoC Mining, Heuristics, PELT Algorithm	Heuristics based, iterative selection
ML Based Models created	No	Yes	Model for determining dependency changes and migration commits
Artifacts created	Scale for developer's activity assessment	No	Dataset of projects

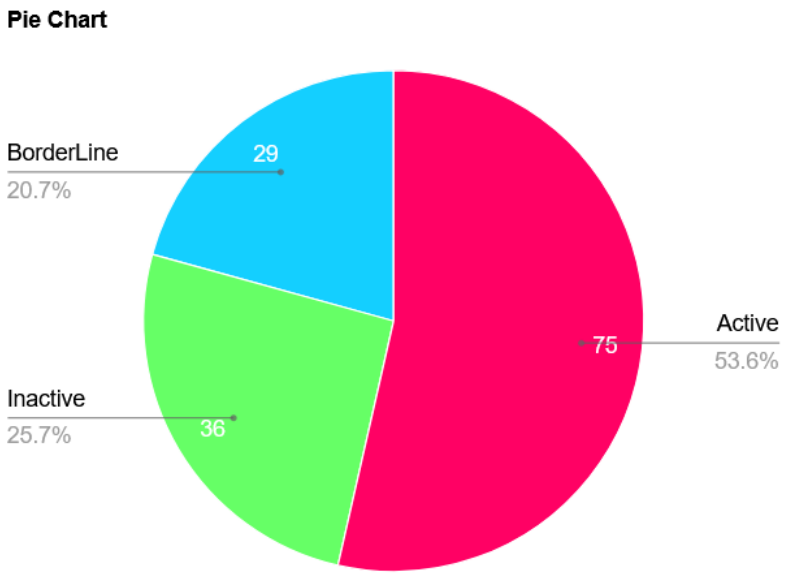

Figure 1. Graphical Visualization of Developer's Activity

Discussions

World of Code provide a mechanism of analysis and understanding of software ecosystem evolution. This infrastructure contains information about various aspects of author and its meta-data linking projects to author's. In the present data, information from the file A_metadata_U is extracted which is present within mongodb database.

An assessment is drawn of the developer's activity in form of commits undertaken. The number of commits done is used to categorize or scale developer as 'active', 'inactive', 'borderline'. This categorization can help in deciding different roles which may be offered to developer's. The number of commits undertaken can also indicate the developer's reputation.

Conclusions

This paper proposed a system to extract the developer's information from WoC infrastructure using mongodb. The methodology involved connecting to the WoC infrastructure and extracting the necessary information from the WoC's mongo database. The entries from A_metadata.U collection from WoC were extracted to a csv file and then scale were proposed on them. The current work can be extended to create a model for predicting developer's metrics more accurately.

References

1. Y. Ma, C. Bogart, S. Amreen, R. Zaretzki and A. Mockus, "World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data," *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, Montreal, QC, Canada, 2019, pp. 143-154, https://doi.org/10.1109/MSR.2019.00031.
2. T. Fry, T. Dey, A. Karnauch and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," *2020 IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*, Seoul, Korea, Republic of, 2020, pp. 518-522, https://doi.org/10.1145/3379597.3387500
3. J. Walden, N. Burgin and K. Kaur, "An Exploratory Study of Project Activity Changepoints in Open Source Software Evolution," *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, Madrid, Spain, 2021, pp. 624-626, https://doi.org/10.1109/MSR52588.2021.00088
4. He, Hao, Runzhi He, Haiqiao Gu, and Minghui Zhou. "A large-scale empirical study on Java library migrations: prevalence, trends, and rationales." In Proceedings of the 29th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp. 478-490. 2021. https://doi.org/10.1145/3468264.3468571
5. Y. Ma, A. Mockus, R. Zaretzki, R. Bradley and B. Bichescu, "A Methodology for Analyzing Uptake of Software Technologies Among Developers," in *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 485-501, 1 Feb. 2022, https://doi.org/10.1109/TSE.2020.2993758
6. H. He, Y. Xu, Y. Ma, Y. Xu, G. Liang and M. Zhou, "A Multi-Metric Ranking Approach for Library Migration Recommendations," *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Honolulu, HI, USA, 2021, pp. 72-83, https://doi.org/10.1109/SANER50967.2021.00016
7. B. L. Sousa, M. A. S. Bigonha, K. A. M. Ferreira and G. C. Franco, "A Time Series-Based Dataset of Open-Source Software Evolution," *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, Pittsburgh, PA, USA, 2022, pp. 702-706, https://doi.org/10.1145/3524842.3528492.
8. Tapajit Dey, Bogdan Vasilescu, and Audris Mockus. 2020. An Exploratory Study of Bot Commits. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, 61–65. https://doi.org/10.1145/3387940.3391502
9. Sayyadnejad, Mohammad Mahdi, Ali Asgari, Ashkan Sami, and Hooman Tahyori. "Exploring the Black Box: Analyzing Explainable AI Challenges and Best Practices Through Stack Exchange Discussions." (2024). https://doi.org/10.21203/rs.3.rs-4304124/v1
10. D. Reid and A. Mockus, "Applying the Universal Version History Concept to Help De-Risk Copy-Based Code Reuse," *2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, Bogotá, Colombia, 2023, pp. 1-12, https://doi.org/10.1109/SCAM59687.2023.00012.

11. Rosa, Giovanni, Antonio Mastropaolo, Simone Scalabrino, Gabriele Bavota, and Rocco Oliveto. "Automatically Generating Dockerfiles via Deep Learning: Challenges and Promises." In *2023 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pp. 1-12. IEEE, 2023. https://doi.org/10.48550/arXiv.2303.15990
12. C. Wang, Z. Chen and M. Zhou, "AutoML from Software Engineering Perspective: Landscapes and Challenges," *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, Melbourne, Australia, 2023, pp. 39-51, https://doi.org/10.1109/MSR59073.2023.00019
13. A. Abdellatif, M. Wessel, I. Steinmacher, M. A. Gerosa and E. Shihab, "BotHunter: An Approach to Detect Software Bots in GitHub," *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, Pittsburgh, PA, USA, 2022, https://doi.org/10.1145/3524842.3527959
14. Jahanshahi, Mahmoud, and Audris Mockus. "Dataset: Copy-based Reuse in Open Source Software." In Proceedings of the 21st International Conference on Mining Software Repositories, pp. 42-47. 2024.
15. Dey, Tapajit, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. "Detecting and characterizing bots that commit code." In *Proceedings of the 17th international conference on mining software repositories*, pp. 209-219. 2020. https://doi.org/10.1145/3379597.3387478
16. S. Amreen, A. Karnauch and A. Mockus, "Developer Reputation Estimator (DRE)," *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, San Diego, CA, USA, 2019, pp. 1082-1085. https://doi.org/10.1109/ASE.2019.00107
17. Valiev, Marat. "External Factors in Sustainability of Open Source Software." PhD diss., Carnegie Mellon University, USA, 2021.