RapidRelease - A Dataset of Projects and Issues on Github with Rapid Releases

Saket Dattatray Joshi, Sridhar Chimalakonda
Department of Computer Science & Engineering
Indian Institute of Technology, Tirupati
Tirupati, India
{tce15b020, ch}@iittp.ac.in

***Abstract*—In the recent years, there has been a surge in the adoption of agile development model and continuous integration (CI) in software development. Recent trends have reduced average release cycle lengths to as low as 1-2 weeks, leading to an extensive number of studies in release engineering. Open-source development (OSD) has also witnessed a rapid increase in release rates, however, no large dataset of open-source projects exists which features high release rates. In this paper, we introduce the RapidRelease dataset, a data showcase of high release frequency open-source projects. The dataset hosts 994 projects from Github, with over 2 million issue reports. To the best of our knowledge, this is the first dataset that can facilitate researchers to empirically study release engineering and agile software development in open-source projects with rapid releases.**

***Keywords*-Github repositories; agile development; release engineering; release cycles; open source development;**

I. Introduction

When asked about rapid release cycles in open source development, a developer said, "The main reason for rapid releases is to deliver value to our users as quickly as possible, and to collect user feedback early and often". A significant amount of research has focused on studying the effect of reduced release cycle lengths [1][2]. There has been an increased industry-driven interest in release engineering and on the other hand, the number of empirical studies, especially in the context of OSD are quite limited [3]. Two of the major facets behind reduced release cycle lengths are the adoption of the agile development philosophy [4] and CI [5].

A literature review of empirical studies on agile development [6] notes that few studies have performed large-scale empirical analysis. A large number of surveys and human-centered studies have been performed on the perceptions of the agile software development philosophy [7][8][9]. Other empirical studies have primarily been on industrial datasets. Similarly, continuous integration has been extensively employed and has become a standard in software development [5]. The influence of CI is even more pronounced in open-source development since the revisions can be released much faster than in industry, and bugs resolved in real-time. Although CI and agile development practices are increasingly implemented in OSD environments, to the best of our knowledge, no large scale dataset facilitating the study of such characteristics exists. Consequently, there is a need for a dataset of open-source projects featuring high release rates.

Github has become the de-facto platform for hosting a majority of open-source repositories. It has been extensively mined for software engineering research. Kalliamvakou et al. discuss that mining Github should be done carefully, considering the various possible perils [10]. They also note that the majority of repositories hosted are personal or inactive, and only a fraction use pull requests. A dataset of active repositories is thus important for research. They also highlight that Github mining is a research gold-mine if filtration of repositories is done properly. In [11], the authors discuss the importance of diversity and representativeness in software engineering research. However, exercising strong precaution and filtering projects for every study is highly effort intensive and desirable for empirical research.

In this paper, we introduce the RapidRelease dataset that consists of 994 projects featuring rapid releases. We identify repositories that have a rapid release cycle of 5-35 days and collect them in our dataset. The core contribution of this paper is a rich open-source dataset for agile software development and release engineering analysis. RapidRelease is publicly available along with documentation, scripts and dataset[1].

The rest of the paper is structured as follows. Section 2 describes the objectives which motivate the methodology; followed by an outline of the dataset construction process and some statistics. Section 3 provides an overview of the dataset structure and Section 4 discusses potential research directions with the dataset. This is followed by future improvements of the dataset in Section 5 and conclusions in Section 6.

II. DATASET CONSTRUCTION

In this section, we first outline the objectives that guide the dataset construction methodology, followed by a description of the dataset extraction and analysis process. The data is mined using GitHub API v3, with python being the primary language.

A. Objectives

The development philosophy of *release early, release often* forms an underpinning characteristic of our dataset [6] [2]. Our ultimate objective for the RapidRelease dataset is a set of open source projects identified to have a high release frequency. The projects should have sufficient development

[1]https://github.com/saketrule/RapidRelease

2574-3864/19/$31.00 ©2019 IEEE
DOI 10.1109/MSR.2019.00088

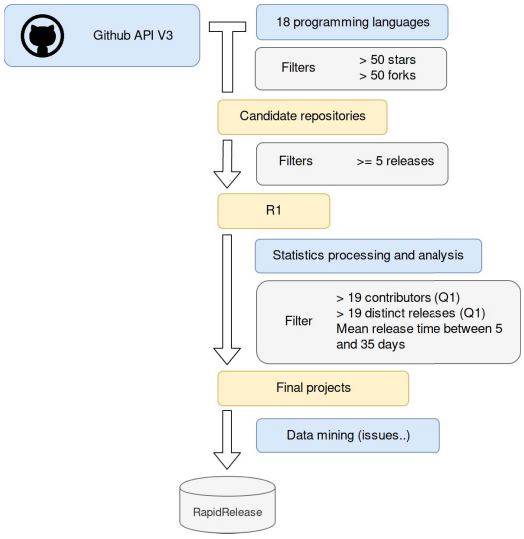

Fig. 1. Dataset construction process overview.

or maintenance history, be collaboratively maintained, and possess characteristics which would enable research into agile development and release engineering. The dataset was extracted and refined in the three stages explained below. The methodology is visualized in Figure 1.

B. Extracting Projects from Github

Stage 1: The first step is to obtain a collection of candidate Github projects. We achieve this by searching for repositories using the Github search API. Similar to [12], we search for repositories from top 18 programming languages. The languages are listed in Table III (not in any specific order). To ensure popular projects, the projects are collected in order of the default Github ranking mechanism and satisfying a threshold of more than 50 stars and more than 50 forks. Similar to [12], our motivation in selecting these metrics is their relation to the project popularity. Our rationale behind the chosen threshold values is to identify and eliminate unsuitable candidates early on to prevent unnecessary data retrieval, processing and storage.

We collect up to a thousand repositories satisfying these criteria for each programming language to form our base consideration pool of 11980 unique repositories. We then collect information on releases, filter out projects with less than 5 releases and discard the remaining repositories. This criteria significantly reduces the number of repositories in consideration, and removes repositories which do not have any releases. The threshold is chosen manually and in apprehension of the more meticulous filtering performed in Stage 2 based on the number of releases. Repositories using more than one language which qualified through the process are counted only once to yield a set of 4341 projects. The collection of projects that are obtained after Stage 1 are labeled as R1.

TABLE I
QUARTILE METRICS FOR R1.

	R1			
Metric[a]	Quartile 1	Quartile 2	Quartile 3	Mean
Contributors	19	48	112.75	87.7
Distinct releases	19.0	35.0	54.75	42.38
Total Time [b]	439.0	837.0	1264.0	884.16
Mean release time	17.0	24.0	31.0	23.72
Median release time	10.0	14.0	20.0	15.60

[a]All time metrics are in days
[b]Total time between the first and last release, in units of days

C. Mining Github

Stage 2: In this step, we extract further data about the repositories, including details about releases and collaborators of the project. We calculate metrics related to release times for each repository in R1. We carefully filtered and inspected the data to deter any specious results. For example, we identified several stagnant releases being published within two days of each other and thus labeled them as non-distinct releases.

As a next step, we discard repositories which have less than 19 contributors (1st Quartile) or 19 distinct releases (1st Quartile). Table I presents some relevant metrics of projects in R1. The final step is filtering repositories with a high release frequency. We follow the rapid release criteria, that is based on agile development principles [4], to retain repositories with a mean distinct release time between 5 and 35 days. This criteria is specifically derived based on our observations from extreme programming and sprint cycles [13]. The process finally yielded a set of 994 projects with rapid releases.

Stage 3: In this stage of the process, we mine issue reports and further details of the selected repositories in Stage 2; and store them in a format described in the next section. Although it is possible to store the entire project data, we specifically extract issue reports to facilitate the study of issues and bug resolution between releases. We do not provide issue comments for data due to the large size and limited support for fast extraction provided by the Github API. However, we provide data links to all information, such as pull requests and issue comments for easy retrieval.

D. Statistics and analysis

Table II presents important statistics of repositories in RapidRelease. Our final dataset contains 994 repositories, with over 2 million issue reports. The final dataset contains projects from 17 languages, *R* being the exception. It is noteworthy that *Javascript*, *Go* and *PHP* are the top 3 languages by number of projects. Table III presents language based distribution of repositories and issue reports in RapidRelease.

As shown in Table II, the dataset features a fast average distinct release frequency of 22 days, maintained over an average of 976 days and 48 distinct releases. A histogram presenting the distribution of projects with mean distinct release time is shown in Figure 2. The large number of open source projects which maintain low average release time is a good insight from our dataset to proliferate further research. The projects also feature a high average of contributor and

TABLE II
STATISTICS OF FINAL DATASET.

Number of projects	994
Avg number of issues	2365
Avg contributors	112
Avg distinct releases	48
Avg total time[a]	976
Mean time between releases[b]	22
Number of issue reports	2351072

[a]Average time between the first and last release, in days
[b]Metric measures average time between distinct releases

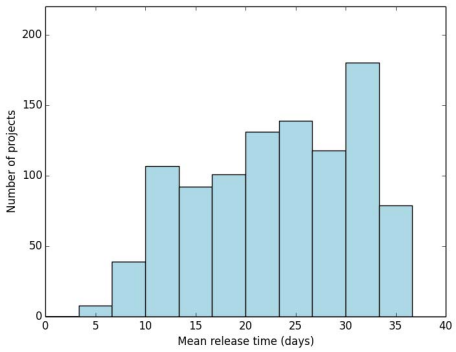

Fig. 2. Histogram of project mean release times in RapidRelease.

issue count numbers, indicating a collaborative nature. All averages mentioned are calculated over repositories.

III. DATASET STRUCTURE

The main dataset is available both in the form of an SQL dump and in CSV format.

A. SQL format

Figure 3 shows the schema design of the SQL dump. The database consists of 4 main tables:

- The *repositories* table contains details of all repositories.
- The *issues* table contains open and closed issues from all repositories.
- The *releases* table contains release details.
- The *developers* table contains user details which take the role of repository owners, issue reporters, or release authors.

Field *repo_id* in tables *releases* and *issues* are foreign keys to repository id. In all tables, field *id* is a primary key.

B. CSV format

The CSV format of dataset contains all data in the SQL dump. The data stored in CSV format also includes data links, meta-data, and JSON-formatted fields. Dataset statistics and list of repositories extracted in the extraction process are also available. The directory structure is described next.

1) In the */releases* and */issues* folders, there is a CSV file for each repository in the final dataset. The file names follow the convention *owner_username*|*repository_name*.csv

TABLE III
LANGUAGE BASED DISTRIBUTION OF REPOSITORIES IN RAPIDRELEASE.

Language	Number of issues	Number of repositories
C	99336	48
Clojure	711	3
Java	147551	69
Scala	50638	40
Python	192138	70
Swift	58065	50
Javascript	568405	196
Viml	3424	2
C++	208357	67
Perl	6010	2
Lua	15472	12
Objective-C	42350	23
R	0	0
Haskell	6163	7
C#	177908	86
Go	379438	157
PHP	298902	123
Ruby	96204	39
Total[a]	2351072	994

[a]Total counts projects using multiple languages only once

2) CSV files with details of repositories in the final dataset are located in the root directory.

IV. POTENTIAL RESEARCH WITH DATASET

In paper [10], the authors discuss how a high number of repositories are for personal use and inactive, making it critical to retrieve a dataset of active repositories for empirical research. Our dataset has been carefully collected and provides attributes such as fast release frequencies, high contributors count and diversity in programming languages, which makes it suitable for a wide range of use cases. The RapidRelease dataset provides several research opportunities, and more importantly, provides a standard dataset for comparative analysis and benchmarking for release engineering studies.

Below, we highlight a few insights and research opportunities that can be leveraged through the dataset.

- **Release Engineering** - An important application of our dataset is enabling research to help analyze and understand patterns in software releases. Our dataset will allow researchers to address important questions relevant to release engineering [14] such as, *What changes in between releases?*, *How many, and what types of issues are raised and resolved?*, *How long does it take for an issue to be resolved, integrated and released?* in an open-source development environment. We have ongoing work which explores some of these questions in the domain of open source development using the dataset.
- **Agile Software Development** - The agile development philosophy has been embraced by much of the industry and open source community, yet there is a distinct lack of studies exploring relevant characteristics in open source projects. A potential application of our dataset can be towards exploring the viability of fast development and release cycle model in OSD. Another motivating topic is the social structure or distribution within the contributor community of a project [15].

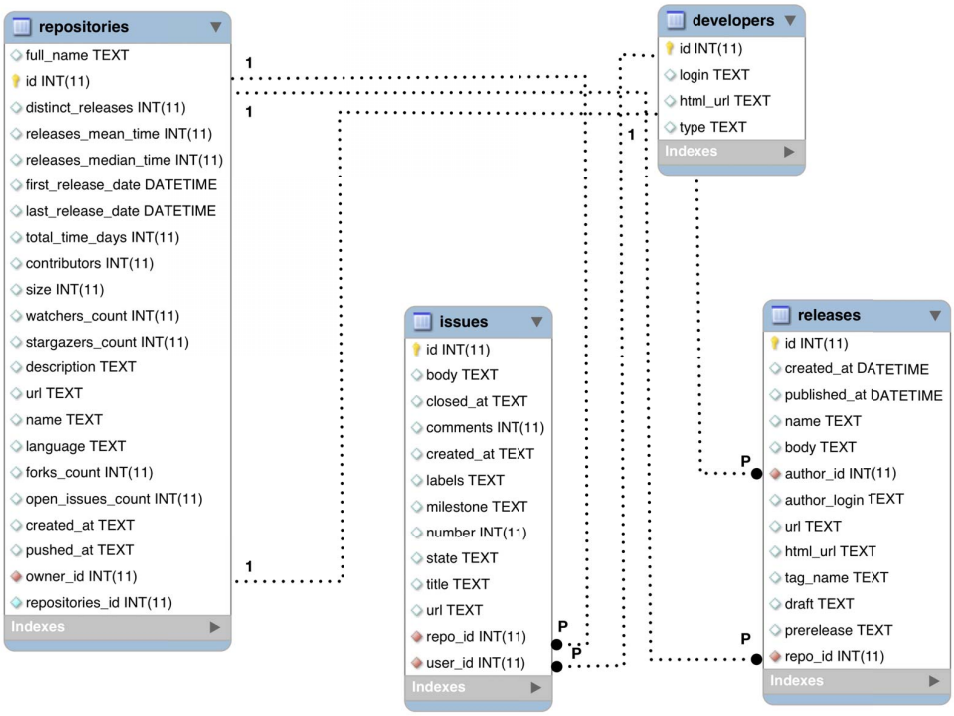

Fig. 3. SQL Database Schema.

- **Further Empirical Studies** - Additionally, the dataset could be potentially used to conduct empirical research to understand code quality, reuse, design documentation, API usage and other topics in software engineering in the context of rapid release projects.

V. LIMITATIONS & FUTURE IMPROVEMENTS

Below, we list the key limitations and scope for future improvements of the dataset.

- There is scope for extending the dataset by incorporating other data, such as pull-requests and issue comments.
- As open-source development evolves, the list of repositories may need to be revised.
- The dataset could be augmented to ensure conformity to further agile principles along with the rapid release criteria. We expect further refined techniques to identify agile characteristics in open-source projects in the future.
- A similar methodology could be used to create datasets specific to a domain or programming language.
- The repository filtration process may not be sufficient to ensure that the projects are being actively developed or maintained. Stars and fork metrics alone may not be indicative of interest or popularity of a repository [16].
- Our methodology should be extended to include issues from external issue trackers along with public issues available on Github.

VI. CONCLUSIONS

Owing to the rise of open source development, agile methodology, and continuous integration in the software development industry, there is a strong need to study the effects of rapid releases. However, the distinct lack of a dataset featuring repositories with high release rates, motivated us to present the RapidRelease dataset in this paper. The dataset features 994 projects with an average of 48 distinct releases, and a mean time of 22 days between releases. We believe that our dataset provides immense opportunities for researchers to perform extensive empirical research and further analysis in release engineering and agile software development.

REFERENCES

[1] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström, and K. Petersen, "On rapid releases and software testing: a case study and a semi-systematic literature review," *Empirical Software Engineering*, vol. 20, no. 5, pp. 1384–1425, 2015.

[2] T. Karvonen, W. Behutiye, M. Oivo, and P. Kuvaja, "Systematic literature review on the impacts of agile release engineering practices," *Information and Software Technology*, vol. 86, pp. 87–100, 2017.

[3] B. Adams and S. McIntosh, "Modern release engineering in a nutshell–why researchers should care," in *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, vol. 5. IEEE, 2016, pp. 78–90.

[4] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," 2001.

[5] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation*. Addison-Wesley Boston, 2011.

[6] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[7] A. Begel and N. Nagappan, "Usage and perceptions of agile software development in an industrial context: An exploratory study," in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. IEEE, 2007, pp. 255–264.

[8] O. Salo and P. Abrahamsson, "Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum," *IET software*, vol. 2, no. 1, pp. 58–64, 2008.

[9] B. Murphy, C. Bird, T. Zimmermann, L. Williams, N. Nagappan, and A. Begel, "Have agile techniques been the silver bullet for software development at microsoft?" in *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013, pp. 75–84.

[10] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[11] M. Nagappan, T. Zimmermann, and C. Bird, "Diversity in software engineering research," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM, 2013, pp. 466–476.

[12] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 155–165.

[13] G. van Valkenhoef, T. Tervonen, B. de Brock, and D. Postmus, "Quantitative release planning in extreme programming," *Information and software technology*, vol. 53, no. 11, pp. 1227–1235, 2011.

[14] A. Dyck, R. Penners, and H. Lichter, "Towards definitions for release engineering and devops," in *Release Engineering (RELENG), 2015 IEEE/ACM 3rd International Workshop on*. IEEE, 2015, pp. 3–3.

[15] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang, "Network structure of social coding in github," in *Software maintenance and reengineering (csmr), 2013 17th european conference on*. IEEE, 2013, pp. 323–326.

[16] H. Borges and M. T. Valente, "Whats in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.