The Relationship Between Folder Use and the Number of Forks: A Case Study on Github Repositories

Jiaxin Zhu, Minghui Zhou
School of Electronics Engineering and Computer Science, Peking University
Key Laboratory of High Confidence Software Technologies, Ministry of Education
Beijing 100871, China
{zhujiaxin, zhmh}@pku.edu.cn

Audris Mockus
Avaya Labs Research
233 Mt Airy Rd, Basking Ridge, NJ
audris@avaya.com

ABSTRACT

Every software development project uses folders to organize software artifacts. We would like to understand how folders are used and what ramifications different uses may have. In this paper we study the frequency of folders used by 140k Github projects and use regression analysis to model how folder use is related to the extent of forking. We find that the standard folders, such as *document*, *test*, and *example*, are not only among the most frequently used folders, but their presence in a project increases the chances that a project code will be forked (i.e., used by others), and increases the number of such forks. This preliminary study of folder use suggests the opportunities to quantify (and improve) file organization practices in large collections of repositories.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Metrics

General Terms

Measurement

Keywords

mining software repository, file organization, folder use, Github

1. INTRODUCTION

Organization of source code and other files in software projects helps the variety of project participants to narrow down the options when searching for information or implementing a change. Participants may never meet as, for example, in open source software (OSS) project, therefore it's important to organize files and folders to enable sharing with peers, successors, collaborators, users, etc., and even, possibly, to attract participants and followers. Small improvements in file organization can deliver great benefits [6]. The extant literature extensively studies personal file organization on computers [1, 7, 2] in order to improve information retrieval by human, but we are not aware of studies targeting collective folder use as in OSS projects.

Github has become the most popular software project hosting site[1], covering a variety of projects from complex operating systems to simple libraries. In this study, we conduct a preliminary study on 140k Github projects[2] to understand the frequencies of folder use among the projects. First, we investigate the most commonly used folder names and find that standard naming scheme is often practiced. For example, *lib*, *src*, *test*, *doc*, and *examples* are among top twenty most commonly used folders, suggesting that different projects tend to follow a convention to create their folders and organize their files. Second, we study the impact of the folder use on the chances that a project gets forked[3]. The more forks a project has, the more likely the project code is used by others and, therefore, there are higher chances to attract potential contributors and (or) followers. By fitting regression models, we find whether the use of these standard folders is associated with higher chances that a project will be forked.

The rest of this paper is structured as follows: We introduce the methodology in section 2. Section 3 analyzes the data and presents the results. We expound the possible threats to the validity of the results in Section 4. Section 5 reviews the related work. We conclude in Section 6.

2. METHODOLOGY

We explore the folder use of the Github projects by mining their version control system (VCS) repositories. We follow procedures to analyze the VCS data described in [9], in particular, we iterate over the following steps: first we retrieve the raw data and perform cleaning, then determine the question to be answered, create measures, perform analysis of these measures, and finally validate the results.

2.1 Context

Github is a software project hosting site built on the distributed version control system, Git. Github projects maintain their artifacts in Git repositories. Each modification (commit) to the code, including who, when, and the complete file path, is recorded.

Forking or making a user's own copy of the original repository, is a common activity in Github work flow [8], which

[1]http://readwrite.com/2011/06/02/github-has-passed-sourceforge
[2]https://github.com/
[3]https://help.github.com/articles/fork-a-repo

ESEM '14 Torino, Italy

leads to multiple repositories for a project. Developers can contribute to the repositories they have forked from by sending pull requests for their local changes to the original repository, or start their own projects based on the repositories they fork.

Github users can subscribe to "watch" a repository[4]. The watcher is notified of new commits, pull requests, and issues. For example, there have been 2379 watchers of Linux kernel[5], who are notified of new features or bug fixes. Among them, 432 forked the repository, while the others may be simply interested in updates to the project [4].

2.2 Retrieve Raw Data

We collected 832,408 repositories from Github in 2012. The meta data of the repositories were also downloaded, which describe the attributes (e.g., create time, repository size, etc.) of a Github repository. We regard these repositories as a representative sample of a broad range of software projects. This is similar to the idea of [10], in which 20,000 projects monitored by Ohloh.net are used as the delegate of all software.

For simplicity, we consider each repository as a separate project. On one hand, it is possible that a large project has several repositories to maintain its different parts. For these cases, the repositories are bound to the lower level projects, sub-projects. On the other hand, it is uncommon that several projects to be maintained in one Git repository.

2.3 Determine Research Questions

On Github, a repository can be forked by any user for various purposes. The major purpose of forking is to provide a way for contributors to write and commit code [8]. Each contributor makes his own fork of the primary repository and publishes his work in the forked repository. The project maintainers integrate the contributions from those forked repositories to the primary repository by "pulling" changes. In addition, a fork can also be used to reuse the code for building an independent project [5]. For example, Linux kernel has been forked more than one thousand times in Github, with some used for application development on Linux.

People who want to learn the code, customize or evolve the software may benefit from the forks. The more forks a repository has, the more likely that there will be more contributors or (and) followers.

In short, the forks of a repository imply that there are additional participants or that the code is reused. We, therefore, conjecture that the folder use may impact the chances of a project having a fork and the number of such forks. In particular: *What are the most commonly used folders in the projects? Are they associated with the chance of the project code being forked?*

2.4 Construct Folders

To study how projects use folders, we exclude projects that are forks of other projects (i.e., that have the same folder structure as the original project in the sample) to ensure there are no duplicated projects in our analysis. 146,124 distinct repositories remained after the filtering.

For each repository, we construct a folder set by extracting folders used in the repository. Each folder is identified by its textual name, e.g., "src". We recognize that folders that have the same purpose may be named in a variety of ways. We use synonyms and similar spellings of the words to recognize which folders could be grouped together as serving the same purpouse. For example, Table 1 gives the names of folders used for testing, documentation, and examples. We consider all folder names used for the same purpose to be of the same type.

Table 1: Names of the Standard Folders

Folder type	Instances
document	doc, docs, document, documents
test	test, tests, t, spec
example	example, examples
Note: case is ignored here, e.g., doc == Doc	

3. RESULTS

We calculate folder frequency across all the projects in Section 3.1, and study the relationship between the folder use and the number of forks in Section 3.2.

3.1 Most Commonly Used Folders

For each folder, we count the number and the proportion of repositories that contain it. Figure 1 shows the distribution of frequencies. Table 2 shows the most frequently used folders. We can see that the majority of folders are only used in a small portion of projects, and the most frequently used folders such as *test* are used in around one third of the projects.

Among the frequently used folders, two typical patterns present themselves. Some folders are related to programming languages and application domains. For example, *com* is a popular international domain name, coming from Java convention. Java maintainer, Oracle, suggests projects using the domain names to name their package folders separating their code from others to avoid name conflict. For example, CSS (cascading style sheets) is a widely used technology in building UI of the web application domain. Projects with Javascript, PHP, and Python source code tend to have *css* folder.

Meanwhile, some appear to be standard folders, especially *test*, *doc* and *examples*. Document and example folders often indicate that a project provides documents describing the requirements, features, functionalities of the project, and provides examples [6]. For example, XORP, an OSS routing project on Github, was recommended for its clean code and good documentation by OpenSourceRouting.org[6]. To contribute to an OSS repository, test is considered to be an essential step[7]. Test folder provides the test code for quality assurance, and may help others to test the code they contribute.

The use of these folders suggests a practice of following conventional folder structure in these projects. It appears that appropriate conventions benefit projects, as Maven guide says, "having a common directory layout would allow for users familiar with one Maven project to immediately feel at home in another Maven project. The advantages are analogous to adopting a site-wide look-and-feel."

Therefore it is of interest to understand if the use of these folders has an impact on the number of forks for a repository,

[4]https://help.github.com/articles/watching-repositories
[5]https://github.com/torvalds/linux
[6]http://conference.apnic.net/__data/assets/pdf_file/0020/50771/osr_apnic34_1346132140.pdf
[7]http://www.sitepoint.com/8-simple-steps-for-contributing-to-open-source/

Table 2: Top 20 Frequently Used Folders

Folder	Proportion	Folder	Proportion
src	0.440	tests	0.145
lib	0.335	plugins	0.144
test	0.321	main	0.137
css	0.217	resources	0.135
js	0.188	java	0.131
templates	0.182	com	0.127
doc	0.176	app	0.120
config	0.175	examples	0.119
views	0.167	controllers	0.118
bin	0.165	models	0.118

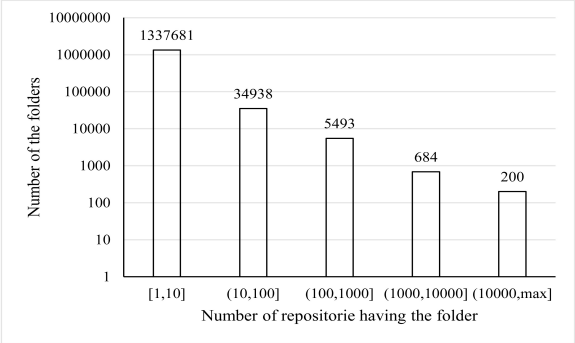

Figure 1: Distribution of Folder Repository Count

and on the chance of the project code being used by others, either contributing to the original project or starting a new one.

3.2 Impact of Folder Use

We fit regression models to investigate the relationship between the number of forks and the use of standard folders. In particular, we use test, document and example to present a basic structure for file organization.

The number of watchers, a factor representing interest in the project, is believed to have a strong correlation with the number of forks [11]. It may help potential contributors to gauge the general interest in a project, and thus determine if they should participate [4]. In other words, the number of forks may be affected by the number of watchers. We, therefore, consider two models: one adjusted for the number of watchers and another without such adjustement.

First, we observe a very large number of projects that are never forked (Table 3, row: true if fork number>0, false otherwise; col: true if watcher number>0, false otherwise). We, therefore, fit a logistic regression model to check if the chances of at least one fork depend on the presence of the standard folders as shown in Equation 1 and 2. The results are shown in Table 4 and 5. Our predictors include the presence of individual folders and the interactions (a simultaneous presence of a subset of folders, such as, "test:example"). We find that, the chances of a fork when adjusted for the number of watchers, are only mediated by the presence of document and test folders, but without the adjustment all three types of folders and their interactions have a significant except for test:example.

Table 3: # of Projects with forking/watching

	FALSE	TRUE
FALSE	2145	106514
TRUE	22	34216

$$Fork\_Num > 0 \sim has\_Document \times has\_Test \times has\_Example \quad (1)$$

$$Fork\_Num > 0 \sim log(Wathch\_Num + 1) + has\_Document \times has\_Test \times has\_Example \quad (2)$$

Table 4: Odds of forking

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.5019	0.0107	-140.22	0.0000
doc	0.2683	0.0274	9.80	0.0000
test	0.4406	0.0162	27.24	0.0000
example	0.4542	0.0364	12.47	0.0000
doc:test	-0.1378	0.0345	-3.99	0.0001
doc:example	-0.1472	0.0681	-2.16	0.0307
test:example	-0.0773	0.0465	-1.66	0.0964
doc:test:example	0.1683	0.0786	2.14	0.0323

Table 5: Odds of forking adjusted for watchers

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-4.8515	0.0245	-198.35	0.0000
log(watch_num+1)	2.6278	0.0150	174.81	0.0000
doc	0.1059	0.0365	2.90	0.0037
test	0.0879	0.0220	3.99	0.0001
example	-0.0307	0.0508	-0.60	0.5453
doc:test	-0.0051	0.0471	-0.11	0.9130
doc:example	0.0702	0.0944	0.74	0.4571
test:example	-0.0978	0.0671	-1.46	0.1450
doc:test:example	0.1062	0.1111	0.96	0.3390

Second, we look at a subset of projects with at least one fork to investigate what affects the number of forks. We fit regression models as described Equation 3 and Equation 4, and the results are shown in Table 6 and 7. We find that, in comparison to the probability of a fork, the number of forks depends on the presence of all three types of folders. The presence of example folder by itself if we adjust for the number of watchers is not associated with the increased number of forks.

$$log(Fork\_Num) \sim has\_Document \times has\_Test \times has\_Example \quad (3)$$

$$log(Fork\_Num) \sim log(Wathch\_Num + 1) + has\_Document \times has\_Test \times has\_Example \quad (4)$$

Table 6: Number of forks

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.2639	0.0096	132.28	0.0000
doc	0.1446	0.0239	6.05	0.0000
test	0.2446	0.0141	17.40	0.0000
example	0.1873	0.0310	6.03	0.0000
doc:test	-0.0565	0.0297	-1.90	0.0572
doc:example	-0.1687	0.0575	-2.93	0.0034
test:example	0.0632	0.0389	1.63	0.1041
doc:test:example	0.1532	0.0657	2.33	0.0198

The models we fit show how the use of folder document, test, and example are associated with the increased chances of forking (Table 4) and with increased number of forks for projects that were forked at least once (Table 6). The interactions (the presence of a combination of these folders) does not appear to affect either the chances of forking or the

Table 7: Number of forks adjusted for watchers

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.3559	0.0110	-32.32	0.0000
log(watch_num+1)	0.8285	0.0045	186.03	0.0000
doc	0.0594	0.0168	3.52	0.0004
test	0.0302	0.0100	3.03	0.0025
example	-0.0479	0.0219	-2.18	0.0290
doc:test	0.0005	0.0210	0.02	0.9821
doc:example	-0.0422	0.0406	-1.04	0.2979
test:example	0.0669	0.0274	2.44	0.0148
doc:test:example	0.0970	0.0464	2.09	0.0363

number of forks (p-val>0.01) except for doc:test decreasing chances and and doc:example decreasing the number. The fraction of deveance explained by these models is fairly small: ...

The degree of interest in the project measured via the number of watchers appears to be a very strong predictor of forking odds and number. Once adjusted for the the degree of interest, only the presence of doc and test folders increase both the odds of forking and the number of forks. The fraction of deviance explained also increases dramatically to ...

The lack of importance of the example folder after the adjustment for the degree of interest is, possibly, related to the fact that the example folder may, in the first place, indicate a successful attempt from project creators to attract project use by others. In other words, the number of watchers and the presence of the example folder may both be reflecting the desire of the project to attract external attention. As a result, only one of these predictors is necessary to explain the increase in the number of forks.

4. THREATS TO VALIDITY

The first threat involves the representativeness of the data set, which determines to what extent our results could be generalized. We conducted our study on 140k Github projects we could access by 2012. The diversity of the projects is guaranteed by Github's popularity and activeness, and the number far surpasses the sample standard used to measure the representativeness of projects [10].

Second, to locate the folders of document, test, example, we enumerate the names we could find as shown in Table 1. These names may not cover all the cases. However, considering the frequency (higher than 0.1%) of the names we cover, it's unlikely the left names (much less than 0.01%) will endanger the results.

Apart from the threats above, we assume that each project has only one repository. Project is identified at different levels of granularity in literature. It should be noted that our results pertain to repository level.

5. RELATED WORK

There are extensive efforts investigating personal file management on computers. Majority of them observe computer users managing their files in field and summarize management patterns, e.g., Barreau et al. [1] found that Macintosh users used subdirectories to organize information and the DOS users did not. Jones et al. [7] found folder hierarchies frequently reflected a tension between organizing information for current use vs. repeated re-use. There are also works measuring the folder structure impact on file navigation. Bergman et al [2] found that Retrieval time and success depended on folder size and depth. Compared to personal information management studies, this paper focuses on collective information management.

There have been attempts that use folder structure to measure the characteristics of software product. For example, Capiluppi et al. [3] studied how folder structure of 25 OSS projects evolved to understand the evolving complexity of the system, Zhou and Mockus [12] used directory structure to measure the modularity of the code. Different from these studies, this paper investigates the commonly used folders and their impact on project code use.

6. CONCLUSIONS

We investigate the most commonly used folders in Github projects, and find many projects follow the conventions to organize files. For example, almost half of the projects use folder *src*, and one third use *lib* and *test*. We find that the use of folders document, test and example has a positive correlation with the number of forks if we don't adjust for the number of watchers.

First, this finding appears to support "Convention over configuration", a software design paradigm which seeks to decrease the number of decisions that developers need to make. The standard (most common) structure may help new project participants immediately to orient themselves. Second, the results suggest that the use of conventional folders may have an impact on the chance of the project code being used by others, who either contribute to the original project or start a new one.

However, "Many open source projects face significant challenges generating and maintaining high quality, end-user documentation"[8], we would like to understand how that could be improved in the future.

7. REFERENCES

[1] D. Barreau and B. A. Nardi. Finding and reminding: File organization from the desktop. *SIGCHI Bull.*, 27(3):39–43, July 1995.

[2] O. Bergman, S. Whittaker, M. Sanderson, R. Nachmias, and A. Ramamoorthy. The effect of folder structure on personal file navigation. *Journal of the American Society for Information Science and Technology*, 61(12):2426–2441, 2010.

[3] A. Capiluppi, M. Morisio, and J. F. Ramil. The evolution of source folder structure in actively evolved open source systems. In *10th International Symposium on Software Metrics*, pages 2–13, 2004.

[4] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Leveraging transparency. *Software, IEEE*, 30(1):37–43, 2013.

[5] K. H. Fung, A. Aurum, and D. Tang. Social forking in open source software: An empirical study. In *CAiSE Forum*, pages 50–57, 2012.

[6] S. Henderson. *How do people manage their documents?: an empirical investigation into personal document management practices among knowledge workers*. PhD thesis, ResearchSpace@ Auckland, 2009.

[7] W. Jones, A. J. Phuwanartnurak, R. Gill, and H. Bruce. Don't take my folders away!: Organizing personal information to get things done. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1505–1508, New York, NY, USA, 2005.

[8] http://oss-watch.ac.uk/resources/archived/documentation

[8] N. McDonald and S. Goggins. Performance and participation in open source software on github. In *CHI'13*, pages 139–144. ACM, 2013.

[9] A. Mockus. Software support tools and experimental work. In V. Basili and et al, editors, *Empirical Software Engineering Issues: Critical Assessments and Future Directions*, volume LNCS 4336, pages 91–99. Springer, 2007.

[10] 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*, pages 466–476. ACM, 2013.

[11] K. Peterson. The github open source development process. Technical report, Technical report, Mayo Clinic, 2013.

[12] M. Zhou, A. Mockus, and D. Weiss. Learning in offshored and legacy software projects: How product structure shapes organization. In *ICSE Workshop on Socio-Technical Congruence*, Vancouver, Canada, May 19 2009.