The Secret Life of Hackathon Code
Where does it come from and where does it go?

Ahmed Imam
University of Tartu
Estonia
ahmed.imam.mahmoud@ut.ee

Tapajit Dey
Lero—the Irish Software Research
Centre, University of Limerick
Limerick, Ireland
tapajit.dey@lero.ie

Alexander Nolte
University of Tartu, Estonia
Carnegie Mellon University,
Pittsburgh, PA, USA
alexander.nolte@ut.ee

Audris Mockus
University of Tennessee
Knoxville, TN, USA
audris@utk.edu

James D. Herbsleb
Carnegie Mellon University
Pittsburgh, PA, USA
jdh@cs.cmu.edu

Abstract—Background: Hackathons have become popular events for teams to collaborate on projects and develop software prototypes. Most existing research focuses on activities during an event with limited attention to the evolution of the code brought to or created during a hackathon. Aim: We aim to understand the evolution of hackathon-related code, specifically, how much hackathon teams rely on pre-existing code or how much new code they develop during a hackathon. Moreover, we aim to understand if and where that code gets reused, and what factors affect reuse. Method: We collected information about 22,183 hackathon projects from DEVPOST– a hackathon database – and obtained related code (blobs), authors, and project characteristics from the WORLD OF CODE. We investigated if code blobs in hackathon projects were created before, during, or after an event by identifying the original blob creation date and author, and also checked if the original author was a hackathon project member. We tracked code reuse by first identifying all commits containing blobs created during an event before determining all projects that contain those commits. Result: While only approximately 9.14% of the code blobs are created during hackathons, this amount is still significant considering time and member constraints of such events. Approximately a third of these code blobs get reused in other projects. The number of associated technologies and the number of participants in a project increase reuse probability. Conclusion: Our study demonstrates to what extent pre-existing code is used and new code is created during a hackathon and how much of it is reused elsewhere afterwards. Our findings help to better understand code reuse as a phenomenon and the role of hackathons in this context and can serve as a starting point for further studies in this area.

Index Terms—Hackathon, Code Reuse, Repository Mining, Commits, Blob Reuse

I. INTRODUCTION

Hackathons are time-bounded events during which individuals form – often ad-hoc – teams and engage in intensive collaboration to complete a project that is of interest to them [1]. They have become a popular form of intense collaboration with the largest collegiate hackathon league alone reporting that their events attract more than 65,000 participants each year[1]. The success of hackathons can at least partially be attributed to them being perceived to foster learning [2], [3], [4] and community engagement [5], [6], [7], [8] and tackle civic, environmental and public health issues [9], [7], [10] which led to them consequently being adopted in various domains including (higher) education [2], [11], [12], (online) communities [6], [13], [14], [15], entrepreneurship [16], [17], corporations [1], [18], [19], [20], and others.

Most hackathon projects focus on creating a prototype that can be presented at the end of an event [21]. This prototype often takes the form of a piece of software. The creation of software code can, in fact, be considered as one of the main motivations for organizers to run a hackathon event. Scientific and open source communities, in particular, organize such events with the aim of expanding their code base [22], [23]. It thus appears surprising that the evolution of the code used and developed during a hackathon has not been studied yet, as revealed by a review of existing literature.

In order to address this gap, we aim to study the evolution of the code used and created by the hackathon team members from two main perspectives. First, we study from where the code *originates*: While teams will certainly develop original code during a hackathon, it can be expected that they will also utilize existing (open source) code as well as code that they might have created themselves prior to the event.

Second, to understand the impact of hackathon code, i.e. code created during a hackathon event by the hackathon team in the hackathon project repository, we aim to study whether and how it *propagates* after the event has ended. There are studies on project continuation after an event has ended [24], [18]. These studies, however, mainly focus on the continuation of a hackathon project in a corporate context [18] and on antecedents of continuous development activity in the same repository that was utilized during the hackathon [24]. The question of where code that has been developed during a hackathon potentially gets reused outside of the context of the original hackathon project has not been sufficiently addressed.

Moreover, we aim to understand what factors might in-

[1]https://mlh.io/about

1

fluence hackathon code reuse, which can be useful for hackathon organizers and participants to foster the impact of the hackathon projects they organize/participate in. These factors would also be of interest to the open source community in general in order to effectively tap into the potential of hackathons as a source of new software code creation.

To cover these two perspectives, we conducted an archival analysis of the source code utilized and developed in the context of 22,183 hackathon projects that were listed in the hackathon database DEVPOST[2]. To track the origin of the code that was used and developed by each hackathon project and study its reuse after an event has ended we used the open source database WORLD OF CODE [25], [26] which allows us to track code usage between repositories. Overall, we looked at over 8.5M blobs [3], over 3M of which were code blobs, as identified with the help of the GITHUB *linguist* [4] tool.

Our findings indicate that around 9.14% of the code blobs in hackathon projects are created during an event, which is significant considering the time and team member constraints. Teams tend to reuse a lot of existing code, primarily as in the form of packages/frameworks. Many of the projects we studied focus on front-end technologies – JavaScript in particular – which appears reasonable because teams often have to present prototypes at the end of an event, which lends itself to UI design. Approximately a third of code blobs created during events get reused in other projects. The number of associated technologies and the number of participants in a project increase the code reuse probability.

In summary, we make the following contributions in the paper: we present an account of code reuse both by hackathon projects and of the code generated during hackathons based on a large-scale study of 22,183 hackathon projects and 1,368,419 projects that reused the hackathon code. We tracked the origins of the code used in hackathon projects, in terms of when it was created and by whom, and also its reuse after an event. We also identified a number of project characteristics that can affect hackathon code reuse. **The replication package for our study is available at** [27].

II. RESEARCH QUESTIONS

As mentioned in section I, the goal of this study is to understand the evolution of hackathon code and identify factors that affect code reuse for these projects.

Our first research question thus addresses the origin of hackathon code:

RQ$_1$. *Where does the code used in hackathon projects originate from?*

Delving deeper into this question, we aim to understand how much of the code used in a hackathon project was actually created *before* the event and reused in the project, how much of the code was developed *during* the hackathon, and, since the projects sometimes continue even after the official end date

of the hackathon, how much of the code was created *after* the event. This leads us to the sub-question:

RQ$_{1a}$. *When was the code created?*

We also aim to understand how much of the code in a hackathon project repository is created by one of the participants, how frequently they reused code created by someone they worked with earlier, and how much of code was created by someone else, leading us to the sub-question:

RQ$_{1b}$. *Who were the original creators of the code?*

Our second research question focuses on the aspect of hackathon code reuse. As noted in section I, existing studies do not address the question of whether and where hackathon code gets reused after an event has ended. However, knowing the answer to this question would be crucial for understanding the impact of hackathons on the larger open source community. Some might perceive hackathons as one-off events where people gather and create some code that is never used again, while in fact they might have an impact on the wider scene of software development and create something of value that transcends individual events. Moreover, it is important to assess in which scale of project hackathon code gets reused (i.e. small projects with few developers and stars or larger projects). This aspect would be useful in understanding the impact of hackathons in greater detail, since, arguably, code that gets reused in larger projects can be perceived to have more impact on the software development community than code that is reused in smaller projects. This leads us to also asking the following second research question:

RQ$_2$. *What happens to hackathon code after the event?*

Finally, our third research question focuses on understanding how different characteristics of a hackathon project can influence the probability of hackathon code reuse. While code reuse in Open Source Software is a topic of much interest, there are only a few studies covering this topic. Moreover, existing studies, e.g. [28], [29], [30], [31] only focus on between 10 and a few hundred projects. For this study, we examined 22,183 hackathon projects, which makes it reasonable to assume that insights from this study – despite them being drawn from hackathon projects only – would add to the existing knowledge about code reuse in general. Thus, we present our third and final research question as:

RQ$_3$. *How can certain project characteristics influence hackathon code reuse?*

Related to this third research question, we formed the following hypotheses that focus on aspects which can reasonably be expected to foster code reuse:

H1 Familiarity: Projects that are attempted by larger teams will have a higher chance of their code being reused, simply because more people are familiar with the code. Moreover, hackathon events that are co-located offer participants more possibilities for interaction which can contribute to a better understanding of each other's code, higher code quality, and consequently foster code reuse.

H2 Prolificness: Code from projects involving many different technologies is more likely to be reused, since: (a) they tend to

---

[2]https://devpost.com/

[3]A blob is a byte string representing a single version of a file, see https://git-scm.com/book/en/v2/Git-Internals-Git-Objects for more details

[4]https://github.com/github/linguist

have more general-purpose code than more focused projects, which affects code reuse as discussed by Mockus [32], and (b) they have a cross-language appeal, opening more possibility for reuse. Similarly, projects with more amount of code created before and during the event (we can not use the code created after the event, for preventing data leakage) should have a higher chance of code reuse by virtue of simply having more code.

H3 Composition: The project composition, i.e. how many blobs in a project are actually related to code, and how many are related to, e.g., data, documentation, or others could be another factor that might influence code reuse. This relationship is likely to be non-linear though, e.g., since we are considering code reuse, a higher percentage of code in the project should increase the probability of reuse, but only up to a certain point, since code from a repository containing only code and no documentation is not very likely to be reused.

III. BACKGROUND

In this section we will situate our work in the context of prior research on hackathon code (section III-A) before discussing existing studies on code reuse (section III-B).

A. Research on hackathon code

The rise in popularity of hackathon event has led to an increased interest to study them [33]. Current research however mainly focuses on the event itself studying how to attract participants [13], [34], how to engage diverse audiences [35], [9], [36], how to integrate newcomers [5], how teams self-organize [37] and how to run hackathons in specific contexts [8], [1], [2]. These studies acknowledge the project that teams work on as an important aspect. The question of where the software code that teams utilize for their project comes from and where it potentially gets reused after an event has not been a strong focus though.

There are also studies that focus on the continuation of software projects after an event has ended [38], [16], [18], [39]. These studies however mainly discuss how activities of a team during, before, and after a hackathon can foster project continuation [18], how hackathon projects fit to existing projects [38], and the influence of involving stakeholders when planning a hackathon project on its continuation [16], [39]. They do not specifically focus on the code that is being developed as part of a hackathon project.

Few studies have also considered the code that teams develop during a hackathon [24], [14]. These studies however mainly focus on code availability after an event [14] or on how activity before and after an event within the same repository that a team utilized during the hackathon can affect reuse [24]. The question of whether and to what extend teams utilize existing code and whether and where the code that they develop during a hackathon gets reused aside from this specific repository has not been addressed.

B. Code reuse

Code reuse has been a topic of interest and is generally perceived to foster developer effectiveness, efficiency, and reduce development costs [40], [28], [30]. Existing work so far mainly focuses on the relationship between certain developer traits [28], [40], [31] and team and project characteristics such as team size, developer experience, and project size and code reuse [41]. Moreover, the aforementioned findings are mainly based on surveys among developers, thus covering their perception rather than actual reuse behavior. In contrast, we aim to study actual code reuse behavior.

There is also existing work that focuses on studying the reuse of the code itself. These, however, are often small scale studies of a few projects [29], [42] focusing on aspects such as automatically tracking reuse between two projects [29] and identifying reasons why developers might choose reuse over re-implementation [42]. In contrast, our aim is to study how the code created during a hackathon evolves i.e. where it comes from and whether and where it gets reused.

Large scale studies on code reuse have been scarce. The few existing studies often focus on code dependencies [43] or on technical dept induced based on reuse [30] which are both not a strong focus for us because our aim is rather to study where hackathon code gets reused. There are studies that discuss the reuse of code on a larger scale [32] and showed that it is mainly code from large established open source projects that get reused, while we aim to study reuse of code that has been developed by a small group of people during a short-term intensive coding event.

IV. METHODOLOGY

A. Data Sources

While hackathon events have risen in popularity in the recent past, many of them remain ad-hoc events, and thus data about those events is not stored in an organized fashion. However, DEVPOST is a popular hackathon database that is used by corporations, universities, civic engagement groups and others to advertise events and attract participants. It contains data about hackathons including hackathon locations, dates, prizes and information about teams and their projects including the project's GITHUB repositories. Organizers curate the information about hackathons and participants indicate which hackathons they participated in, which teams they were part of and which projects they worked on. DEVPOST does not conduct accuracy checks.

However, DEVPOST does not contain all the information required for answering our research questions. We thus leveraged the WORLD OF CODE dataset for gathering additional information about projects, authors, and code blobs. WORLD OF CODE is a prototype of an updatable and expandable infrastructure to support research and tools that rely on version control data from the entirety of open source projects that use Git. It contains information about OSS projects, developers (authors), commits, code blobs, file names, and more. WORLD OF CODE provides maps of the relationships between these entities, which is useful in gathering all relevant information required for this study. We used version S of the dataset for the analysis described in this paper which contains repositories identified until Aug 28, 2020.
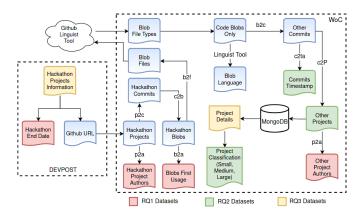
Fig. 1. **Data Collection Workflow: Highlighting the different data sources used and the process of gathering the required information from them, and the data used in answering our research questions**

B. Data Collection and Cleaning

Here we describe how we collected the data required for answering our research questions, along with details of all the filtering we introduced. An overview of the approach is shown in fig. 1, which also highlights the different data sources and what data was used for answering each research question.

1) Selecting appropriate hackathon projects for the study: We started by collecting information about 60,479 hackathon projects from DEVPOST. Since the project ID used in DEVPOST is different from the project names in WORLD OF CODE, in order to link these hackathon projects to the corresponding projects in WORLD OF CODE, we looked at the corresponding GITHUB URLs, which could be easily mapped to the project names used in WORLD OF CODE, where the project names are stored as GitHubUserName_RepoName. After filtering out the projects without a GITHUB URL, we ended up with 23,522 projects. While trying to match these projects with the corresponding ones in WORLD OF CODE, we were not able to match 1,339 projects, which might have been deleted or had their names changed afterwards. Thus, we ended up with 22,183 projects for further analysis.

2) Gathering the contents (blobs) of the project: Our first step was to identify all code blobs used in the hackathon projects. WORLD OF CODE does not have a direct map between projects and blobs, so we started by collecting commits for all hackathon projects using the project-to-commit (*p2c*) map in WORLD OF CODE, which covers all commits for each hackathon project. For the 22,183 hackathon projects, we collected 1,659,435 commits generated in the hackathon repository. Then, we gathered all the blobs associated with these commits using the commit-to-blob (*c2b*) map in WORLD OF CODE, which yielded 8,501,735 blobs, which are all the blobs associated with the hackathon projects under consideration.

3) Filtering to only select code blobs: The hackathon project repositories, like most other OSS project repositories, have more than just code in them — they also contain images, data, documentation, etc. Since our aim in this project is the identification of the reuse of "code", we decided to filter the

blobs to only have the ones related to "code". In order to achieve that, we looked at the filenames for each of the blobs in the project (since blobs only store the contents of a file, not the file name) using the blob-to-filename (*b2f*) maps in WORLD OF CODE. After that, we used the *linguist* tool from GITHUB to find out the file types. The *linguist* tool classifies files into types "data, programming, markup, prose", with files of type "programming" being what we are focusing on in this study. Additionally, we marked all the files that are not classified by the tool as files of type "Other", with the presumption that they do not contain any code. Therefore, we focused only on the blobs whose corresponding files are classified as "programming" files by the *linguist* tool, which reduced the number of blobs under consideration to 3,079,487.

4) Gathering data required to identify the origins of hackathon code (RQ1): To address our first research question, we needed information about the first commits associated with each of the 3,079,487 blobs under consideration. Fortunately, it is possible to get information about the first commit that introduced each blob using WORLD OF CODE. We extracted the *author* of that first commit, along with the *timestamp*, which would be useful in identifying when the blob was first created. We have the end date for each of the hackathon events from DEVPOST, however, it does not include any information about the start date of the hackathon. We consider the start of a hackathon 72 hours before the end date. This assumption appears reasonable since hackathons are commonly hosted over a period of 48 which are often distributed over three days [44], [16]. We also conducted a manual investigation of 73 randomly selected hackathons and found that only 2 projects (2.7%) were longer than 3 days, which empirically suggested that our assumption would be valid for most of the hackathons under consideration. Under this assumption, we have the start and end dates of each of the hackathon events, and we used that information to identify if a blob used in a hackathon project was created before, during, or after the hackathon event.

We can identify the first commit that introduced the hackathon blobs under consideration, the *author* of that commit, and all of the developers who have been a part of the hackathon project [5] using WORLD OF CODE. With this data, we can determine if the blob was first created by a member of the hackathon team or someone else. In order to dig further and understand if the blob was created in another project a member of the hackathon project participated in, we used WORLD OF CODE to identify the project associated with the first commit for each blob under consideration, identified all developers of that project, and checked if any of them are members of the team that created the hackathon project under consideration. This lets us identify if the blob was created by (a) a developer who is a participant of the hackathon project (thus, they are creating the code/reusing what they had created earlier), (b) a developer who was part of a project one of the

---

[5] We used the approach outlined by [45] for author ID disambiguation to merge all of the different IDs belonging to one developer together, which is a common occurrence, as discussed in [46]

Hypothesis	Variable	Variable Description	Source	Value Range (min-max)	Median
H1: Familiarity	no.Participant	Number of Participants in the hackathon	DEVPOST	2 - 10	3
	is.colocated	hackathon is held in single or multi location	DEVPOST	TRUE: 12,445 (97%) FALSE:436 (3%)	
H2: Prolificness	no.Technology	Number of different technologies the hackathon is related to	DEVPOST	1 - 40	5
	Before	Number of blobs in the hackathon project repo that were created before the event	WORLD OF CODE	0 - 205,666	4
	During	Number of blobs in the hackathon project repo that were created during the event	WORLD OF CODE	1 - 3288	23
H3: Composition	pctCode	Fraction of the blobs in the project repo that are classified as "programming"	WORLD OF CODE and GITHUB	0 - 1	0.40
	pctMarkup	Fraction of the blobs in the project repo that are classified as "Markup"	WORLD OF CODE and GITHUB	0 - 0.96	0.02
	pctData	Fraction of the blobs in the project repo that are classified as "Data"	WORLD OF CODE and GITHUB	0 - 0.999	0.12
	pctProse	Fraction of the blobs in the project repo that are classified as "Prose"	WORLD OF CODE and GITHUB	0 - 0.999	0.03

participants of the hackathon project also contributed to (which might suggest that they are familiar with the code, which might have influenced the reuse of that code in the hackathon project), or (c) someone else who has not contributed to any of the projects the hackathon project developers previously contributed to (which would suggest a lack of direct familiarity with the code from the hackathon participants' perspective).

5) Gathering data to identify hackathon code reuse (RQ2): Our second research question focuses on the reuse of *hackathon code*, which, per our definition (see section I), refers to the blobs created during the hackathon event by one of the members of the hackathon team. Therefore, to address this question, we utilized the results of our earlier analysis in order to focus only on the code blobs which satisfy the following two conditions: (a) The blob was first introduced during the hackathon event and (b) the blob was created by one of the hackathon project developers. After identifying 581,579 blobs that met these conditions, we collected all commits containing these blobs from WORLD OF CODE using the blob-to-commit (*b2c*) map, and we collected the projects where these commits are used using the commit-to-project (*c2p*) map. WORLD OF CODE has the option of returning only the most central repositories associated to each commit, excluding the forked ones (based on the work published in [47]), and we used that feature to focus only on the repositories that first introduced these blobs, and excluded the ones that were forked off of that repository later, since most forks are created just to submit a pull request and counting such forks would lead to double-counting of code reuse.

In addition to understanding how the blobs get reused, we also wanted to understand if they are reused in very small projects, or if larger projects also reuse these blobs. So, we needed a way to classify the projects into different categories. We focused on two different project characteristics for the purpose of such classification: the number of developers who contributed to that project, and the number of *stars* it has on GITHUB, a measure available from a database (MongoDB) associated with WORLD OF CODE. Both the number of devel-

opers and *stars* are quintessential measures of project size and popularity and were found to have a low correlation (Spearman Correlation: 0.26), so we decided to use both measures. Instead of manually classifying the projects using these variables using arbitrary thresholds, we decided to use *Hartemink's pairwise mutual information based discretization method* [48], which was applied to a dataset with log-transformed values of the number of stars and developers for projects, to classify them into three categories: Small, Medium, and Large. We found different thresholds for the number of developers and *stars* (for no. of developers, $> 2 \rightarrow$ Medium projects and $> 6 \rightarrow$ Large; for stars, $> 1 \rightarrow$ Medium and $> 14 \rightarrow$ Large), and classified a project as "Large" if it is classified as such by either the number of developers or the number of *stars*, and used a similar approach for classifying them as "Medium". The remaining projects were classified as "Small". Overall, we identified 1,368,419 projects that reused at least one of the 581,579 blobs, and using our classification, 1,220,114 (89.2%) projects were classified as "Small", 116,177 (8.5%) as "Medium", and 32,128 (2.3%) as "Large".

6) Collecting Data for Identifying the factors that affect hackathon code reuse (RQ3): In addition to tracking hackathon code reuse (RQ2), we also aimed to study factors that can affect this phenomenon. For this purpose we collected various characteristics of the hackathon projects, both from DEVPOST and WORLD OF CODE, and extracted the variables of interest, per the hypotheses presented in section II.

The data we collected for RQ2 was for the blobs, so, in order to find out if code from a project were reused, we investigated how many blobs from a project was reused, and calculated the ratio of the number of reused code blobs and the total number of code blobs in the project. This revealed that almost 60% of projects had none of their code reused. So, we decided to pursue a binary classification problem for predicting if a project has at least one code blob reused or not instead of doing regression analysis.

For the purpose of our analysis, we excluded hackathon projects with a single member, since a hackathon project

Fig. 2. **Plot of Who created how much of the Hackathon Code and When**

Fig. 3. **Top 5 languages for blobs created before, during, and after hackathons**

Fig. 4. **Top 5 languages for blobs created by project members, co-contributors, and others**

"team" with a single participant does not really make a lot of sense, and also the projects that were not related to any existing technology, since these likely were non-technical events. By looking for code reuse, we also automatically filtered out any project that had no code blobs in its repository. After these filterings, we were left with 12,881 hackathon projects.

For the variables related to **H3**, the composition of the repository, we have 5 categories, 4 of which are dictated by the GITHUB *linguist* tool: *Code(programming), Markup, Data*, and *Prose*, and a category *Other* for all file types not classified by the tool. We looked at what percentage of the blobs in the projects belonged to which type. Since they all sum up to 100%, 4 of these variables are sufficient to describe the fifth variable. In order to remove the resulting redundancy, we decided to remove the entry for type *Other*, since its effect is sufficiently described by the remaining variables.

The description of all the variables along their sources and values are presented in table I.

7) Analysis Method for Identifying project characteristics that affect code reuse (RQ3): As we noted in the hypotheses presented in section II, we are expecting some of the project characteristics to have a linear effect on hackathon code reuse, while some should have a more complex non-linear effect. The goal of our analysis is not to make the best predictive model that gives the optimum predictive accuracy, instead, we are trying to find out which of the predictors have a significant effect by creating an explanatory model. As noted by Shmueli [49], these two are very different tasks.

In order to achieve our goal of having linear and non-linear predictors in the same model and be able to infer the significance of each of them, we decided to use Generalized Additive Models (GAM). Specifically, we used the implementation of GAM from the `mgcv` package in *R*.

V. RESULTS

Here we will discuss our findings in relation to our research questions and discuss the result of a small case-study on some examples of code reuse for selected hackathon projects.

A. Origins of hackathon code (RQ1)

As mentioned in section II and section IV-B4, we focused on two aspects while looking for the origins of the code in the hackathon project repositories, when was it created (RQ1.a), and who was the original creator of the code blob (RQ1.b). In terms of "when", we examined if the first creation of the code blob under consideration was *Before, During*, or *After* the corresponding hackathon event. In terms of "who",
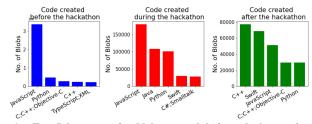
we checked if the first creator of the code blob was one of the members of the hackathon project (*Project Member*), or someone who was a contributor to a project in which one of the members of the hackathon project contributed to as well (*Co-Contributor*), or someone else (*Other Author*).

The result of the analysis is presented in fig. 2, showing that, overall, 85.56% of the code used in the hackathon projects was created before an event. Most of these reused blobs were part of a framework/library/package used in the hackathon project, which aligns with the findings of [32]. Around 9.14% of the blobs were created during events, since participants need to be efficient during an event owing to the time limit [18] which fosters reuse as previously discussed in the context of OSS [28]. We also found that 5.3% of the blobs were created after an event, suggesting that most teams do not add a lot of new content to their hackathon project repositories after the event. This finding is in line with prior work on hackathon project continuation [24].

Looking at top languages for the blobs created at different times (fig. 3), identified by the GITHUB *linguist* tool, we found that most of the code reused by hackathon projects (created before) is JavaScript, and other top languages together indicate that most of the reused code by hackathon projects are related to web development frameworks. JavaScript is also the most common language worked on during the hackathons we studied, followed by Java, Python, Swift, and C#/Smalltalk, indicating that most hackathon projects work on developing web/mobile apps. C++ was the most common language for code developed after the event, followed by Swift and JavaScript, showing a slight shift in the type of work done after the event, favoring Machine Learning applications.

> **RQ1.a**: *85.56% of the code (in terms of the no. of blobs) in the hackathon project repositories is created before the hackathons, with around 9.14% of the code being created during the events (which is significant considering the limited duration of the hackathons).*

Figure 2 shows that, overall, the original creators of most of

the code blobs (69.54%) in the hackathon project repositories are someone who is not a part of the team. They are mostly the original creators of some project/package/framework used by the hackathon team. Around one-third (29.47%) of the code was created by the project members, and the reuse of code from co-contributors in other projects is very limited (0.99%). This aspect has not been extensively studied in the context of work on hackathons yet.

Looking at the top languages for the code created by different authors, as shown in fig. 4, we can see that, once again, most of the code created by developers not part of the hackathon team is JavaScript, which is similar to the code created before the event (fig. 3). This is not surprising, since they have a great deal of overlap (fig. 2). Most of the code created by project members indicate a leaning towards web/mobile app development, and most of the C++ and Python code was found to be related to Machine Learning frameworks.

> **RQ1.b**: *The members of the hackathon teams created around 29.47% of the code blobs, while 69.54% of the code blobs are created by developers outside the team (mostly authors of some project/package/framework used by the team).*

If we consider the code blobs created during and after the hackathon event, as shown by the combined picture in fig. 2, which are the main contributions of the hackathons in terms of code creation, we see that most of that code (97% for the code created during the event, and 93% of the code created after the event) was actually created by the project members. Moreover, we also find that the project members often reuse the code they had written earlier in their projects, 15.67% of all the code belong to this category. This finding is in line with prior work on hackathon projects in that teams often prepare their projects e.g. by setting up a repository [24] and/or making (detailed) plans on what they want to achieve during an event [18].

It is also worth noting that, typically, widely-used frameworks like Django, Rails, jQuery, etc have decades of development history and a large number of contributors. Thus, around 9.14% of blobs being created during the short duration of hackathon events (72 hours per our assumption) by teams of around 2-3 members (up to a maximum of 10 members — see table I), is indeed significant and it highlights the importance of hackathons in generating new code.

> **Origin of the Hackathon Code (RQ1)**: *Hackathon projects often reuse code in terms of some package/framework. Teams also tend to reuse their own code. Most of the code created during or after the event is created by the hackathon team members.*

B. Hackathon code reuse

As discussed in section II and section IV-B5, our goal while looking for hackathon code reuse is twofold: First, we want to see how much of the code gets reused, and second, we want to find if they get reused in small, medium, or large projects. By
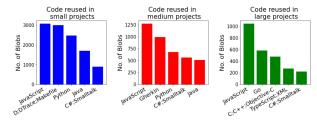

Fig. 5. **Top 5 Languages for the reused code blocks in different projects**

following the procedure outlined in section IV-B5, we found that 167,781 (28.8%) of the 581,579 hackathon code blobs got reused in other projects.

We further classified the projects that reused these code into *Small, Medium*, and *Large*, as discussed in section IV-B5. To recap, 89.2% of the projects that reused the hackathon code blobs were classified as *Small*, 8.5% were *Medium*, and 2.3% were classified as *Large* projects. By investigating the blobs reused by these projects we found that, unsurprisingly, there are a number of instances where a blob was reused in projects of different categories. However, such cases were found to be quite rare, in fact, only 8.85% of reused blobs got reused in more than one project. By looking at the size of the projects a blob was reused in, we found that over half (57.73%) of the blobs are only reused in other *Small* projects, around one-third (32.85%) are reused in *Medium* projects, and less than a tenth (9.42%) are reused in *Large* projects.

The top-5 languages for the blobs reused by various projects are shown in fig. 5. As we can see JavaScript still remains the most common, and Python, C/C++, C#/Smalltalk, Java were among the top ones as well. While most reused blobs are related to web/mobile apps/frameworks, we also found the relatively uncommon Gherkin being the second most common language for *Medium* projects, and the *Small* projects reused a lot of blobs related to D/DTrace/Makefile.

We were interested in exploring the temporal dynamics of code reuse as well. Therefore, we looked at the reuse of hackathon code blobs over-time for the duration of two years (104.3 weeks) after the corresponding hackathon event ended. The result of that analysis is shown in fig. 6, which shows the *weekly* hackathon code reuse for 2 years after the end of the corresponding hackathon event, with the fraction of total number of hackathon code blobs (581,579) reused per week on the Y-axis. As we can see from this plot, while overall 28.8% of the hackathon code blobs were reused, over the span of a single week, no more than 0.8% of the blobs got reused. This finding is in line with prior work on hackathon project continuation (e.g. Nolte et al. [24] found that continuation activity drops quickly within one week after a hackathon before reaching a stable state) within the same repository that the team used during the hackathons. A clear trend of the code reuse dropping and then saturating after some time is visible, which is significant because it indicates that the *code created in the hackathon events continue to bear some value even after 2 years* have passed after the event. For code reuse in *Small* projects, the knee point comes after around 10-15 weeks, while for the *Medium* and *Large* projects, it comes much earlier, in around a month. It is also a bit surprising to see code reuse
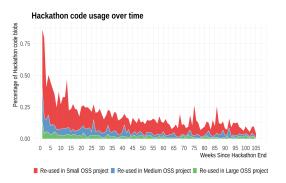
Fig. 6. **Plot depicting hackathon weekly code reuse in projects of different categories over the period of 2 Years**

peaking so soon after the event, but this could be due to the participants of the event putting/influencing people they know to put the code they think is valuable to some other project where they think it might be of use. This distinction has not been studied in prior work on hackathon code.

> ***Hackathon code reuse (RQ2):*** *Around 28.8% of hackathon code blobs got reused in other projects, with 57.73% of the code being used in Small projects, 32.85% in Medium projects, and 9.42% in Large projects. Most of the reused blobs were related to web/mobile apps/frameworks. The temporal dynamics of code reuse show a clear trend of it reducing over time, and then saturating to a stable value.*

C. Characteristics affecting code reuse

Our third and final Research Question was about identifying what project characteristics affect code reuse and we formed three hypotheses about what factors might be affecting it, which were presented in section II. Using the procedure outlined in section IV-B6, we gathered the variables of interest related to the three hypotheses, as presented in table I. As discussed in section IV-B7, we decided to use Generalized Additive Models (GAM) to identify the variables that have a significant impact on code reuse.

Since we presumed the variables related to **H1** and **H2** would have a linear effect on the probability of code reuse for a project, we kept them as linear terms in the model. The variables related to **H3** were presumed to have a non-linear effect, so they were used as non-linear terms in the model. The formula we used for invoking the GAM model was:

$Y \sim$ *no.Participant + is.colocated + no.Technology + Before + During + s(pctProse) + s(pctData) + s(pctCode) + s(pctMarkup)*

The result of the analysis is presented in table II, which shows that all of the variables related to hypotheses **H1** and **H2**, which we assumed would affect code reuse, were indeed significant. The effect direction, communicated by the signs of the estimates for the corresponding variables match the rationale we presented in section II. These findings are partially in line with prior work on hackathon project continuation in that complex projects for which hackathon participants have prepared prior to the hackathon showed

A. Linear Variables (Hypothesis)	Estimate	Std. Error	p-value
no.Participant (H1)	0.2078	0.0181	< 0.0001
is.colocated-**TRUE** (H1)	0.2034	0.1030	0.0483
no.Technology (H2)	0.0261	0.0060	< 0.0001
Before (H2)	0.0001	0.0000	< 0.0001
During (H2)	0.0036	0.0004	< 0.0001
B. Non-Linear Variables (Hypothesis)	**edf**	**p-value**	**Partial Effect Plot**
pctProse (H3)	3.8984	0.0195	
pctData (H3)	3.7148	0.2490	
pctCode (H3)	3.8425	0.0208	
pctMarkup (H3)	6.6779	0.0001	

increased continuation activity [24]. In that study, the team size was however negatively related to project continuation while for the study in this paper we found the reverse to be true for code reuse. One possible explanation for this discrepancy can be related to larger teams having more opportunities and wider networks to spread the news about their project [18].

As for the variables related to **H3**, the "pctData" (see table I for definition) variable was found to be not significant, but the other three variables were. Since the effects of these variables were non-linear in the model, we decided to observe their partial effect plots, which shows the relationship between the two plotted variables (an outcome and an explanatory variable) while adjusting for interference from other explanatory variables [50]. As demonstrated by the partial effect plots, and the effective degrees of freedom associated with each of these variables, each of them actually have a non-linear effect on the response variable.

Let's take a closer look at the effects of the three significant variables related to **H3**. For "pctProse", which refers mostly to documentation files (e.g. Markdown, Text, etc.), we see that having some documentation is good, however, projects which have around half its total content as documentations are not likely to have their code reused. However, rather surprisingly, we see that projects with almost all of their content as documentation are more likely to have their code reused. On closer inspection, we found that these are quite large projects with a lot of data stored in their corresponding repositories in the form of text or other files (An example of such a project is

https://github.com/sreejank/PoliClass). Therefore, though most of them have a good amount of code, by volume, it appears that it is almost all made up of files of type "Prose", which causes this predictor to show positive values for projects very high amount of "Prose" files. Such projects however are common in particular in civic and scientific hackathons where participants often develop projects that are related to utilizing specific datasets (e.g. [5], [37], [14]). Our finding thus can potentially point to a specific use case that is beneficial for code reuse after an event has ended.

On closer inspection of the variable "pctCode", we realized that it refers to how much of the total content in a repository is of type "code", not the absolute number of code blobs, therefore, having a more balanced repository with a good mix of other types of files signal that it is of higher quality, thus increasing the chance of code reuse. This is somewhat expected since for code to be reused it is beneficial to have accompanying documentation as well as use cases (data). As we can observe, projects with over 60% of their content related to code take a big hit when it comes to their code getting reused. This finding can potentially be due to hackathon teams only having a finite amount of time during an event to actually develop code and the more the code they develop the more likely it is that they do not have time to "polish" it for reuse.

The behavior of the "pctMarkup" variable is more complex than the rest (it also has a higher *edf* value), so it is hard to summarize the interaction without a detailed inspection on a larger dataset, however, it looks like having up to around 60% markup content (e.g. HTML, CSS, LaTeX, etc.) can lead to a higher propensity of code reuse. The reason for the increase for projects with a very high percentage of markup content is likely similar to what we observed for the "pctProse" variable.

> ***Characteristics affecting Code Reuse (RQ3):*** *The hypotheses presented in section II were found to hold. All of the variables related to **H1** and **H2** were significant and had effects as anticipated. The effect of the variables related to **H3** were more complex, which led to additional insights about hackathon code reuse.*

D. A Case-Study on Code Reuse

In addition to investigating the research questions presented here, we also conducted a small-scale case study on a few projects selected by stratified random sampling to gain additional insights.

We observed that there are 4 main types of hackathon projects: some containing few blobs that were created before the corresponding hackathon, but have a good amount of activity after the event (Type - **A** : *After*), while some projects had a large number of blobs that were created before the event, but with little activity afterwards (Type - **B**: *Before*). Some projects contained code created before, during, and after the event (Type **C**: *Continuous*), and some mostly contained code created during the event (Type **D**: *During*).

For this case study, we looked at both projects that had some of their code reused, and projects that did not, and chose one project of each type (A, B, C, and D) for both of these categories at random. The names and details about how many blobs each project had, when they were created, and for the blobs that were created *during* the hackathon, what was their type and how many were reused are presented in table III.

Detailed findings for the individual projects with code reuse are listed below:

• *Opportunity-Hack-2015-Arizona_Team1:* This project had one code and one data blob reused, both with content related to python libraries, in 2 and 1 other projects respectively.

• *TheMichaelHu_PickyPusheen*: A number of reuses, mostly icons and configuration from a framework. Only one code blob created by one of the project members was reused, and it was a report for a debug tool they ran.

• *drfuzzyness_WearHax-OlivMatt:* This project shows evidence of code reuse related to Occulus/Kinect/Wii which was widely reused afterwards by projects of different sizes. All these blobs are under the `assets` folder and most probably are part of a framework though, so either this team created that framework, or were the first ones to use it. This project had no README or any other documentation.

• *kylemsguy_building-point:* A good amount of code from this project was used in another small project, likely created by one of the team members. Almost all of the reused files were part of a framework used in the project.

One curious distinction between the projects with and without code reuse was that, in most cases, projects with code reuse had other types of blobs reused as well, and for those without code reuse, they mostly have none of their blobs reused. This is in line with the previously discussed finding that additional documentation and a potential use case can foster code reuse. It is also worth noting that for almost all of the code reuses we observed, the code was part of a package/library/framework used in the project. This is not too surprising, since we are only looking at *exact* code reuse. However, we filtered our dataset to only look for blobs first created by one of the members of the project team during the event, so it is very likely that the framework was created/modified to some extent by the project team members, and that newly created/modified framework was later used by others. At the same time, it is possible that someone might have made the exact same changes to a file as made by a member of the hackathon project, which then gets counted as reuse. Further study is needed to ascertain the probability of such chance events happening by accident.

VI. IMPLICATIONS

Our findings have a number of implications for research and practice. They can serve as a valuable guidance for scientific and other communities that aim to organize hackathons for expanding their existing code base. Organizers could suggest participating teams to attempt projects that do not require developing a large amount of code and rather focus on a specific use case e.g. related to an existing data set. Moreover, they should suggest teams to also spend time on not only developing code but also providing additional materials and documentation, and also for the teams to reuse existing code

TABLE III
DETAILS OF THE PROJECTS SELECTED FOR CASE STUDY: SHOWING NO. OF BLOBS CREATED BEFORE, DURING, AND AFTER THE CORRESPONDING HACKATHON EVENTS, HOW MANY BLOBS OF DIFFERENT TYPES WERE CREATED *during* THE EVENT, AND HOW MANY OF THEM GOT REUSED

Category	Hackathon Project (Type)	Blob Creation Time			Blob Reuse	Blob Types				
		Before	During	After		Other	Data	Markup	Code	Prose
Projects with code usage	Opportunity-Hack-2015-Arizona_Team1 *(Type-A)*	42	177	*316*	Reused	0	1	0	1	0
					Not reused	21	33	19	97	5
	TheMichaelHu_PickyPusheen *(Type-B)*	*3372*	520	1	Reused	66	3	0	9	1
					Not reused	90	259	2	78	12
	drfuzzyness_WearHax-OlivMatt *(Type-C)*	3154	823	1194	Reused	4	60	0	17	2
					Not reused	19	645	0	53	4
	kylemsguy_building-point *(Type-D)*	29	*81*	10	Reused	7	7	0	15	0
					Not reused	0	26	0	26	0
Projects without code usage	quki_IDHACK2016 *(Type-A)*	26	96	*102*	Used	0	1	0	0	0
					Not reused	0	44	0	50	1
	Marblez_Haven-App *(Type-B)*	*448*	20	1	Reused	0	0	0	0	0
					Not reused	1	11	0	7	1
	shkbfzl_hs-lunchbot *(Type-C)*	186	48	59	Reused	0	0	0	0	0
					Not reused	0	2	0	46	0
	jrdbnntt_DaReactTV *(Type-D)*	4	*63*	1	Reused	2	0	0	0	0
					Not reused	2	11	6	41	1

from their existing code base rather than attempting to develop a lot of original code. This approach can in turn improve efficiency and foster code reuse after an event has ended.

With respect to research, our findings provide an initial account on how code gets reused and created during a hackathon as well as whether and where it gets reused afterwards. They indicate that not much new code gets developed during a hackathon and much of the code used by the teams is actually reused from existing code, thus altering our perception that hackathons are intensive code creation events. Moreover, they indicate that hackathon code indeed gets reused and that hackathons can thus be more than one-off coding events.

VII. LIMITATIONS AND THREATS TO VALIDITY

For our study, we tracked the code generation and usage on a blob level - represented in WORLD OF CODE by the SHA1 hash value of each blob - which means that we focused only on exact code reuse since any changes in the file contents would lead to a change in the blob SHA1 value that we used to identify each blob. However, it is quite common to make minor changes in a code file while using it in a different context, and that aspect will not be captured in our study, nor can we capture the reuse of code snippets. Moreover, our analysis does not consider the size of a file since we aree looking at the SHA1 values of the blobs, i.e. our analysis cannot distinguish between reuse of a small file and that of a large file.

The DEVPOST dataset does not include the start date of the hackathon events but it is essential information needed to answer our research questions. We assumed the duration of the hackathons to be 72 hours based on existing literature and a manual investigation of 73 randomly selected hackathons 71 of which lasted up to 3 days. However, that might not have been the case for all of the events we studied which may affect the results of RQ1 and RQ3.

We relied on the GITHUB Linguist tool to categorize files and we only focused on files with type "Programming", how-ever the categorization is not infallible, e.g. the type "Markup" contains HTML and CSS files which could be considered code instead of documentation.

Finally, in our study we only considered hackathon projects, thus, our findings may not be generalizable to other types of software projects and repositories.

VIII. CONCLUSION AND FUTURE WORK

In this study, we investigated the origins of hackathon code and its reuse after an event. We found that most hackathon projects reuse existing code and that code created during events also gets reused by other OSS projects later on. Our study also revealed that project characteristics related to its prolificness and the developers' familiarity with the code positively affects code reuse, and the composition of the project in terms of what file types it contains have an effect as well. In summary, our findings agree with most earlier studies, and reiterate the impact of hackathon events, at the same providing an account of code reuse in Open Source Software.

There are several ways to extend this research, e.g. considering code clones/snippets while looking for code reuse (e.g. by looking at the associated CTAG tokens - a dataset available in WORLD OF CODE), identifying other factors that affect code reuse, including code quality [51], [52], project popularity [53], [54], the type of Open Source license used, etc. Looking deeper into the code created during the hackathons, it might also be interesting to see to what extent the teams use bots [55], [56] which might aid in the understanding of hackathon code reuse as well. We hope that further studies will explore these and other related topics, and give us a clearer understanding of the impact of hackathons and code reuse.

REFERENCES

[1] E. P. P. Pe-Than, A. Nolte, A. Filippova, C. Bird, S. Scallen, and J. D. Herbsleb, "Designing corporate hackathons with a purpose: The future of software development," *IEEE Software*, vol. 36, no. 1, pp. 15–22, 2019.

[2] J. Porras, A. Knutas, J. Ikonen, A. Happonen, J. Khakurel, and A. Herala, "Code camps and hackathons in education-literature review and lessons learned," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[3] A. Fowler, "Informal stem learning in game jams, hackathons and game creation events," in *Proceedings of the International Conference on Game Jams, Hackathons, and Game Creation Events*. ACM, 2016, pp. 38–41.

[4] A. Nandi and M. Mandernach, "Hackathons as an informal learning platform," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 2016, pp. 346–351.

[5] A. Nolte, L. B. Hayden, and J. D. Herbsleb, "How to support newcomers in scientific hackathons-an action research study on expert mentoring," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW1, pp. 1–23, 2020.

[6] D. Huppenkothen, A. Arendt, D. W. Hogg, K. Ram, J. T. VanderPlas, and A. Rokem, "Hack weeks as a model for data science education and collaboration," *Proceedings of the National Academy of Sciences*, vol. 115, no. 36, pp. 8872–8877, 2018.

[7] N. Taylor, L. Clarke, M. Skelly, and S. Nevay, "Strategies for engaging communities in creating physical civic technologies," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 507.

[8] S. Möller, E. Afgan, M. Banck, R. J. Bonnal, T. Booth, J. Chilton, P. J. Cock, M. Gumbel, N. Harris, R. Holland *et al.*, "Community-driven development for computational biology at sprints, hackathons and codefests," *BMC bioinformatics*, vol. 15, no. 14, p. S7, 2014.

[9] A. Hope, C. D'Ignazio, J. Hoy, R. Michelson, J. Roberts, K. Krontiris, and E. Zuckerman, "Hackathons as participatory design: Iterating feminist utopias," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 61.

[10] B. Baccarne, P. Mechant, D. Schuurma, L. De Marez, and P. Colpaert, "Urban socio-technical innovations with and by citizens," *Interdisciplinary Studies Journal*, vol. 3, no. 4, p. 143, 2014.

[11] K. Gama, B. Alencar, F. Calegario, A. Neves, and P. Alessio, "A hackathon methodology for undergraduate course projects," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.

[12] H. Kienzler and C. Fontanesi, "Learning through inquiry: A global health hackathon," *Teaching in Higher Education*, vol. 22, no. 2, pp. 129–142, 2017.

[13] N. Taylor and L. Clarke, "Everybody's hacking: Participation and the mainstreaming of hackathons," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 172.

[14] B. Busby, A. Matthew Lesko *et al.*, "Closing gaps between open software and public data in a hackathon setting: user-centered software prototyping," *F1000Research*, vol. 5, 2016.

[15] R. C. Craddock, D. S. Margulies, P. Bellec, B. N. Nichols, S. Alcauter, F. A. Barrios, Y. Burnod, C. J. Cannistraci, J. Cohen-Adad, B. De Leener *et al.*, "Brainhack: a collaborative workshop for the open neuroscience community," *GigaScience*, vol. 5, no. 1, p. 16, 2016.

[16] D. Cobham, K. Jacques, C. Gowan, J. Laurel, S. Ringham *et al.*, "From appfest to entrepreneurs: using a hackathon event to seed a university student-led enterprise," in *11th annual International Technology, Education and Development Conference*, 2017.

[17] A. Nolte, "Touched by the hackathon: a study on the connection between hackathon participants and start-up founders," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems*, 2019, pp. 31–36.

[18] A. Nolte, E. P. P. Pe-Than, A. Filippova, C. Bird, S. Scallen, and J. D. Herbsleb, "You hacked and now what? -exploring outcomes of a corporate hackathon," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–23, 2018.

[19] M. Komssi, D. Pichlis, M. Raatikainen, K. Kindström, and J. Järvinen, "What are hackathons for?" *IEEE Software*, vol. 32, no. 5, pp. 60–67, 2015.

[20] B. Rosell, S. Kumar, and J. Shepherd, "Unleashing innovation through internal hackathons," in *Innovations in Technology Conference (InnoTek), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[21] M. A. Medina Angarita and A. Nolte, "What do we know about hackathon outcomes and how to support them? - a systematic literature review," in *Collaboration Technologies and Social Computing*. Springer, 2020.

[22] E. P. P. Pe-Than and J. D. Herbsleb, "Understanding hackathons for science: Collaboration, affordances, and outcomes," in *International Conference on Information*. Springer, 2019, pp. 27–37.

[23] A. Stoltzfus, M. Rosenberg, H. Lapp, A. Budd, K. Cranston, E. Pontelli, S. Oliver, and R. A. Vos, "Community and code: Nine lessons from nine nescent hackathons," *F1000Research*, vol. 6, 2017.

[24] A. Nolte, I.-A. Chounta, and J. D. Herbsleb, "What happens to all these hackathon projects? - identifying factors to promote hackathon project continuation," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, pp. 1–26, 2020.

[25] 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," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 143–154.

[26] Y. Ma, T. Dey, C. Bogart, S. Amreen, M. Valiev, A. Tutko, D. Kennard, R. Zaretzki, and A. Mockus, "World of code: Enabling a research workflow for mining and analyzing the universe of open source vcs data," *arXiv preprint arXiv:2010.16196*, 2020. [Online]. Available: https://arxiv.org/pdf/2010.16196

[27] "Replication package," https://github.com/woc-hack/track_hack.

[28] S. Haefliger, G. Von Krogh, and S. Spaeth, "Code reuse in open source software," *Management science*, vol. 54, no. 1, pp. 180–193, 2008.

[29] N. Kawamitsu, T. Ishio, T. Kanda, R. G. Kula, C. De Roover, and K. Inoue, "Identifying source code reuse across repositories using lcs-based source code similarity," in *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2014, pp. 305–314.

[30] D. Feitosa, A. Ampatzoglou, A. Gkortzis, S. Bibi, and A. Chatzigeorgiou, "Code reuse in practice: Benefiting or harming technical debt," *Journal of Systems and Software*, p. 110618, 2020.

[31] G. von Krogh, S. Spaeth, and S. Haefliger, "Knowledge reuse in open source software: An exploratory study of 15 open source projects," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, 2005, pp. 198b–198b.

[32] A. Mockus, "Large-scale code reuse in open source software," in *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 7–7.

[33] J. Falk Olesen and K. Halskov, "10 years of research with and on hackathons," in *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, 2020, pp. 1073–1088.

[34] Y. Hou and D. Wang, "Hacking with npos: collaborative analytics and broker roles in civic data hackathons," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, pp. 1–16, 2017.

[35] L. Paganini and K. Gama, "Engaging women's participation in hackathons: A qualitative study with participants of a female-focused hackathon," in *International Conference on Game Jams, Hackathons and Game Creation Events 2020*, 2020, pp. 8–15.

[36] A. Filippova, E. Trainer, and J. D. Herbsleb, "From diversity by numbers to diversity as process: supporting inclusiveness in software development teams with brainstorming," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 152–163.

[37] E. H. Trainer, A. Kalyanasundaram, C. Chaihirunkarn, and J. D. Herbsleb, "How to hackathon: Socio-technical tradeoffs in brief, intensive collocation," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 2016, pp. 1118–1130.

[38] H. Lapp, S. Bala, J. P. Balhoff, A. Bouck, N. Goto, M. Holder, R. Holland, A. Holloway, T. Katayama, P. O. Lewis *et al.*, "The 2006 nescent phyloinformatics hackathon: a field report," *Evolutionary Bioinformatics Online*, vol. 3, p. 287, 2007.

[39] A. Ciaghi, T. Chatikobo, L. Dalvit, D. Indrajith, M. Miya, P. B. Molini, and A. Villafiorita, "Hacking for southern africa: Collaborative development of hyperlocal services for marginalised communities," in *IST-Africa Week Conference, 2016*. IEEE, 2016, pp. 1–9.

[40] M. Sojer and J. Henkel, "Code reuse in open source software development: Quantitative evidence, drivers, and impediments," *Journal of the Association for Information Systems*, vol. 11, no. 12, pp. 868–901, 2010.

[41] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from stackoverflow: An exploratory study on android apps," *Information and Software Technology*, vol. 88, pp. 148–158, 2017.

[42] B. Xu, L. An, F. Thung, F. Khomh, and D. Lo, "Why reinventing the wheels? an empirical study on library reuse and re-implementation," *Empirical Software Engineering*, vol. 25, no. 1, pp. 755–789, 2020.

[43] D. M. German, "Using software distributions to understand the relationship among free and open source software projects," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 24–24.

[44] A. Nolte, E. P. P. Pe-Than, A.-A. O. Affia, C. Chaihirunkarn, A. Filippova, A. Kalyanasundaram, M. A. M. Angarita, E. H. Trainer, and J. D. Herbsleb, "How to organize a hackathon - a planning kit," *ArXiv*, vol. abs/2008.08025, 2020.

[45] 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," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 518–522. [Online]. Available: https://doi.org/10.1145/3379597.3387500

[46] T. Dey, A. Karnauch, and A. Mockus, "Representation of developer expertise in open source software," *ArXiv*, vol. abs/2005.10176, 2020.

[47] A. Mockus, D. Spinellis, Z. Kotti, and G. J. Dusing, "A complete set of related git repositories identified via community detection approaches based on shared commits," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 513–517. [Online]. Available: https://doi.org/10.1145/3379597.3387499

[48] A. J. Hartemink, "Principled computational methods for the validation discovery of genetic regulatory networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[49] G. Shmueli *et al.*, "To explain or to predict?" *Statistical science*, vol. 25, no. 3, pp. 289–310, 2010.

[50] M. D. Williams, "peplot() for plotting partial effects," https://rpubs.com/milesdwilliams15/328471, accessed: 2021-01-11.

[51] T. Dey and A. Mockus, "Modeling relationship between post-release faults and usage in mobile software," in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 56–65.

[52] T. Dey and A. Mockus, "Deriving a usage-independent software quality metric," *Empirical Software Engineering*, vol. 25, no. 2, pp. 1596–1641, Mar 2020.

[53] T. Dey and A. Mockus, "Are software dependency supply chain metrics useful in predicting change of popularity of npm packages?" in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 66–69.

[54] T. Dey, Y. Ma, and A. Mockus, "Patterns of effort contribution and demand and user classification based on participation patterns in npm ecosystem," in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 36–45.

[55] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, "Detecting and characterizing bots that commit code," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 209–219.

[56] T. Dey, B. Vasilescu, and A. Mockus, "An exploratory study of bot commits," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 61–65.