UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Ahmed Samir Imam Mahmoud

THE SECRET LIFE OF HACKATHON CODE

Master's Thesis (30 ECTS)

LTAT.00.002

Supervisor: Alexander Nolte

Tartu 2021

The Secret Life of Hackathon Code

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:** The aim of the study is 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, I aim to understand if and where that code gets reused, and what factors affect reuse. **Method**: I 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. I 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. I 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:** My 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. the 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.

Keywords:

Hackathon, Code Reuse, Repository Mining, Commits, Blob Reuse

CERCS:

P170 Computer science, numerical analysis, systems, control

Häkatonidel loodud koodi salajane elu

Kokkuvõte:

Taust: Häkatonidest on saanud meeskondade jaoks populaarsed üritused erinevate projektidega töötamiseks ja tarkvara prototüüpide loomiseks. Enamik olemasolevaid uuringuid keskendub sündmuse ajal toimuvale tegevusele, pöörates minimaalselt tähelepanu häkatonil kasutatud või selle käigus loodud programmikoodi e. koodi arengule. **Eesmärk:** Uuringu eesmärk on mõista kuidas kood häkatonidel areneb, täpsemalt seda, kui palju häkatoni meeskonnad loodavad olemasolevale koodile või kui palju uut koodi nad häkatoni jooksul arendavad. Lisaks on eesmärgiks mõista, kas ja kus täpsemalt seda koodi taaskasutatakse ning millised tegurid mõjutavad koodi taaskasutamist. **Meetod:** Kogusin häkatonide andmebaasist, DEVPOST, teavet 22,183 häkatoniprojekti kohta ja hankisin WORLD OF CODE seotud koodi (blob'i e. binaarversiooni), selle autorid ning projekti parameetrid. Uurisin, kas häkatoni projektide koodiplokid loodi enne sündmust, selle ajal või pärast seda, tuvastades algse blob'i loomise kuupäeva ja autori, ning kontrollisin ka, kas algne autor oli häkatonil osaleja. Jälgisin koodi taaskasutamist tuvastades esmalt kõik häkatoni ajal loodud blob'id ning seejärel tegin kindlaks kõik projektid, mis sisaldasid samasid blob'e. **Tulemus:** Kuigi häkatonide ajal luuakse vaid 9,14% koodist, on see siiski märkimisväärne, võttes arvesse häkatonide ajaraami ja osavõttu. Ligikaudu kolmandik loodud koodist taaskasutatakse järgmistes projektides. Seotud tehnoloogiate ja projektis osalejate arv suurendab taaskasutuse tõenäosust veelgi. **Järeldus:** Minu uuring näitab, millises mahus kasutatakse häkatonil juba olemasolevat koodi või luuakse uus kood ning kui palju sellest koodist hiljem taaskasutatakse. Tulemused aitavad paremini mõista koodi taaskasutamist, kui nähtust, ning häkatonide rolli selles kontekstis. Lisaks on töö edasiste uuringute lähtepunktiks selles valdkonnas.

Märksõnad:

Häkaton, programmikoodi taaskasutamine, programmikoodi hoidla kaevandamine, programmikoodi esitamine, blob'ide taaskasutus

CERCS:

P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 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 [50]. 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 [51, 20, 42] and community engagement [45, 28, 57, 41] and tackle civic, environmental and public health issues [26, 57, 4] which led to them consequently being adopted in various domains including (higher) education [51, 22, 32], (online) communities [28, 56, 5, 8], entrepreneurship [7, 43], corporations [50, 47, 33, 52], and others.

Most hackathon projects focus on creating a prototype that can be presented at the end of an event [38]. 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 [49, 55]. 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, I aim to study the evolution of the code used and created by the hackathon team members from two main perspectives. First, I 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. A detailed discussion about the developed concrete research questions and the motivation behind them is explained in section 1.1.

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, I aim to study

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

whether and how it *propagates* after the event has ended. There are studies on project continuation after an event has ended [44, 47]. These studies, however, mainly focus on the continuation of a hackathon project in a corporate context [47] and on antecedents of continuous development activity in the same repository that was utilized during the hackathon [44]. 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, I aim to understand what factors might influence 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, I 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 I used the open source database WORLD OF CODE [36, 37] which allows to track code usage between repositories. Overall, I analyzed over 8.5M blobs [3], over 3M of which were code blobs, as identified with the help of the GITHUB *linguist* [4] tool.

My 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, since most of the hackathon events have a strict time boundaries which in most cases are hosted over a period of 48 which are often distributed over three days [46, 7] and teams are usually formed during the events where people from different backgrounds and skill sets work together for the first time and teams may fluctuate during the hackathon events [9]. Teams tend to reuse a lot of existing code, primarily

[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`

as in the form of packages/frameworks. Many of the projects I 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, I make the following contributions in this thesis: I 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. I 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. I also identified a number of project characteristics that can affect hackathon code reuse. **The replication package for the study is available at** [1].

1.1 Research Goals

As mentioned in the introduction (section 1), the goal of this study is to understand the evolution of hackathon code and identify factors that affect code reuse for these projects.

The first research question addresses the origin of hackathon code:

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

Delving deeper into this question, I 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?*

I 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?*

My second research question focuses on the aspect of hackathon code reuse. As noted in section 1, 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 to also asking the following second research question:
RQ$_2$. *What happens to hackathon code after the event?*

Finally, the 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. [24, 31, 18, 59] only focus on between 10 and a few hundred projects. For this study, I 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, I present the third and final research question as:
RQ$_3$. *How can certain project characteristics influence hackathon code reuse?*

Related to this third research question, I 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 have more general-purpose code than more focused projects, which affects code reuse as discussed by Mockus [39], 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 (I 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 I am 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.

2 Background

In this section, I will situate the work in the context of prior research on hackathon events (section 2.1), hackathon code (section 2.2) before discussing existing studies on code reuse (section 2.3).

2.1 Research on hackathon events

Hackathons are time-bounded events that are sponsered by universities, corporations, civic engagement groups and others where participants form teams to work on a project that makes interest to them. Usually, team members are having different skills, education levels and goals, however, they work together as a team and compete with other teams. Organizers offer an environment that supports the participating teams to pitch, program, and present their actionable ideas and a prototype of a software project. The popularity of hackathons arisen from their growing global occurrence. Hackathons have become an activity for many organizations, universities [51, 22, 32] and large corporations [50, 47, 33, 52] as an approach to encourage innovation and tackle global issues.

Hackathons can be conducted for several reasons like creating software prototypes, product features or bug fixes [38]. It can be also conducted for providing opportunities for learning, networking and sharing ideas [38]. It can be also conducted for encouraging to create startups and Entrepreneurship [38]. The most commonly conducted events though are focusing on creating artifacts like software prototypes, library or functioning software [46]. The aim of my study is to focus on the hackathon's main artifact which is the code produced in such events and provide insights about how the code brought to or created during a hackathon.

2.2 Research on hackathon code

The rise in popularity of hackathon events has led to an increased interest to study them [17]. Current research however mainly focuses on the event itself like studying how to attract participants [56, 27], how to engage diverse audiences [48, 26, 19], how to

integrate newcomers [45], how teams self-organize [58] and how to run hackathons in specific contexts [41, 50, 51]. 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. I aim to address this gap by analyzing the code produced during the hackathon events to get insights from where this code comes from and how this code gets reused after the events in other open-source products.

There are also studies that focus on the continuation of software projects after an event has ended [34, 7, 47, 6]. These studies however mainly discuss how activities of a team during, before, and after a hackathon can foster project continuation [47], how hackathon projects fit to existing projects [34], and the influence of involving stakeholders when planning a hackathon project on its continuation [7, 6]. They do not specifically focus on the code that is being developed as part of a hackathon project. In contrast, I aim to analyze the project characteristics that can affect the code reuse probability.

Few studies have also considered the code that teams develop during a hackathon [44, 5]. These studies however mainly focus on code availability after an event [5] or on how activity before and after an event within the same repository that a team utilized during the hackathon can affect reuse [44]. 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. I aim to address this gap by analyzing the other open-source projects that used the hackathon code and categorizing them into small, medium and large projects based on defined criteria in the methodology section 3.

2.3 Code reuse

Code reuse has been a topic of interest and is generally perceived to foster developer effectiveness, efficiency, and reduce development costs [54, 24, 18]. Existing work so far mainly focuses on the relationship between certain developer traits [24, 54, 59] and team and project characteristics such as team size, developer experience, and project size

and code reuse [3]. Moreover, the aforementioned findings are mainly based on surveys among developers, thus covering their perception rather than actual reuse behavior. In contrast, I 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 [31, 61] focusing on aspects such as automatically tracking reuse between two projects [31] and identifying reasons why developers might choose reuse over re-implementation [61]. In contrast, the 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 [23] or on technical dept induced based on reuse [18] which are both not a strong focus for us because the aim is rather to study where hackathon code gets reused. There are studies that discuss the reuse of code on a larger scale [39] and showed that it is mainly code from large established open source projects that get reused, while I aim to study reuse of code that has been developed by a small group of people during a short-term intensive coding event.

3 Methodology

In this section, I will explain the steps I followed to answer the research questions. My main goal was to design a methodology that can help me to understand how the hackathon code get generated *(RQ1)*, how the hackathon code gets reused in other open-source projects *(RQ2)* and what project characteristics affect code reuse probability *(RQ3)*. I start by identifying the data sources used to collect all the information needed. I then explain about WORLD OF CODE platform that is used heavily to collect most of the information needed to answer the research questions. I then explain the steps followed to collect the data and analyze the results.

3.1 Data Sources

The aim here is to identify the data sources that can be used to collect information about hackathons and in particular hackathon code. While hackathon events have risen in popularity in the recent past, many of them remain ad-hoc events, and thus data about those events are 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 data are mainly filled manually by organizers and participating teams during and after the hackathon events and DEVPOST does not conduct accuracy checks.

My main goal was to get information about the hackathon projects related to the code produced in the project and to track the instances where this code was originally created and when it was used after the hackathon event. Although DEVPOST was the main source to identify the hackathon projects and it let me put my hand on more than 27,000 hackathon project, with information about these projects like the number of participants,

the number of technologies, prizes, winning teams, the event locations, etc.., DEVPOST does not contain all the information required for answering the research questions, especially data related to the project activities like commits, code blobs and authors. I thus leveraged the WORLD OF CODE [5] dataset for gathering additional information about projects, commits, authors, and code blobs. In the next section, I will explain about WORLD OF CODE and the capabilities of this infrastructure in getting the relationship between code files with other projects, commits and authors.

3.2 World Of Code

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. The WORLD OF CODE platform can be considered as a mining platform that collects information about all open source projects like commits, authors, timestamps, etc.. by cloning all repositories and perform data extraction and reorganization [35] and provide basic tools that can help researchers to get insights about the interdependencies of the project artifacts among all FLOSS[6] projects. The WORLD OF CODE provides handy tools with Perl and python interfaces that not only allow to collect of information about commits, blobs, projects, and authors but also collect information about the interdependencies of these artifacts among all FLOSS projects. This will help to get information about the origin of each hackathon blob and how these blobs are reused after the hackathon event. I used version S of the dataset for the analysis described in this paper which contains repositories identified until Aug 28, 2020.

The relations in WOC are represented by three characters like *x2y* where x represents the source entity and y represents the required target entity. There are several entities in WoC which are the base for all the relations. The main entities are Project, Commit, Author, Blob, and File. Figure 1 represents a diagram for all relations between these five entities.

[5] WoC website: `https://worldofcode.org`
[6] Free/Libre Open Source Software

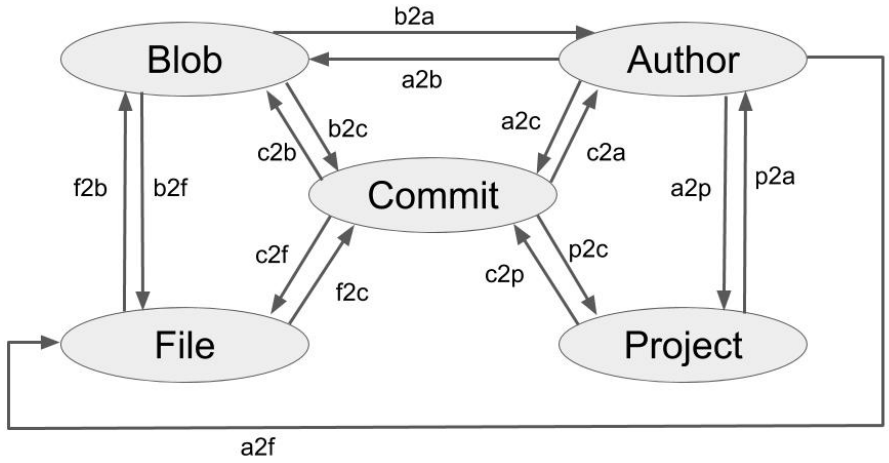


Figure 1. WoC entities and the available maps between them [37]

For instance, If I would like to get all commits for a particular author, so using the *a2c* (Author-To-Commit) map we can get this information. Similarly, *b2a* means to retrieve the first author who created a particular blob. Table 1 represents all available entities and the supported relationships in WOC.

Table 1. List of Entities and relationships

List of Entities	List of relationships
a = Author b = Blob c = Commit cc = Child Commit f = File h = Head Commit ob = Parent Blob p = Project pc = Parent Commit P = Forked/Root Project ta = Time Author td = Tdiff tk = Tokens (ctags) trp = Torvalds Path	a2b,a2c,a2f,a2ft,a2p,a2trp b2a,b2c,b2f,b2ob,b2tk c2b,c2cc,c2f,c2h,c2pc c2p,c2P,c2ta,c2td f2a,f2b,f2c p2a,p2c,P2c td2c,td2f

The data collection in WORLD OF CODE is done through shell and Perl scripts, which is considered the fastest way to collect large about of data. The system also provides a python interface named *Oscar.py* where you can write python scripts to collect the data, however, the python interface is considered slower than Perl. The WORLD OF CODE tutorial and a benchmarking of both Perl and python interfaces were done in the WORLD OF CODE tutorial page [2].

3.3 Data Collection and Cleaning

Here I describe how I collected the data required for answering the research questions, along with details of all the filtering I introduced. An overview of the approach is shown in fig. 2, which also highlights the different data sources and what data was used for answering each research question.

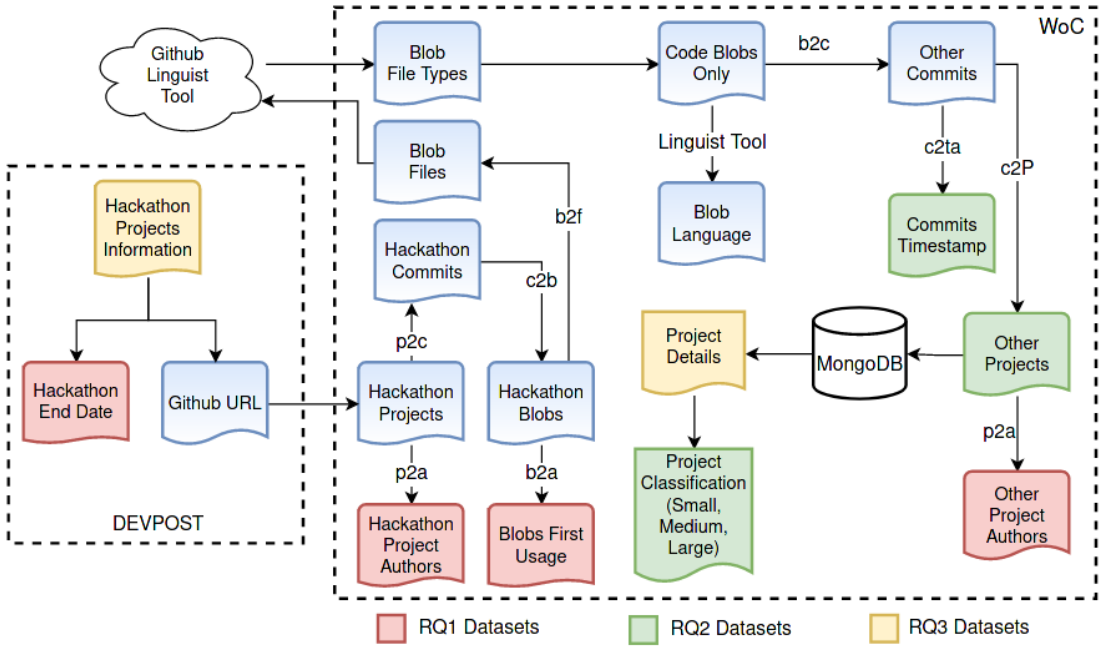


Figure 2. **Data Collection Workflow: Highlighting the different data sources used and the process of gathering the required information from them, and the data used in answering the research questions**

3.3.1 Selecting appropriate hackathon projects for the study

I 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, I 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` and represented in figure 3. After filtering out the projects without a GITHUB URL, I ended up with 23,522 projects. While trying to match these projects with the corresponding ones in WORLD OF CODE, I was not able to match 1,339 projects, which might have been deleted or had their names changed afterward. Thus, I ended up with 22,183 projects for further analysis.

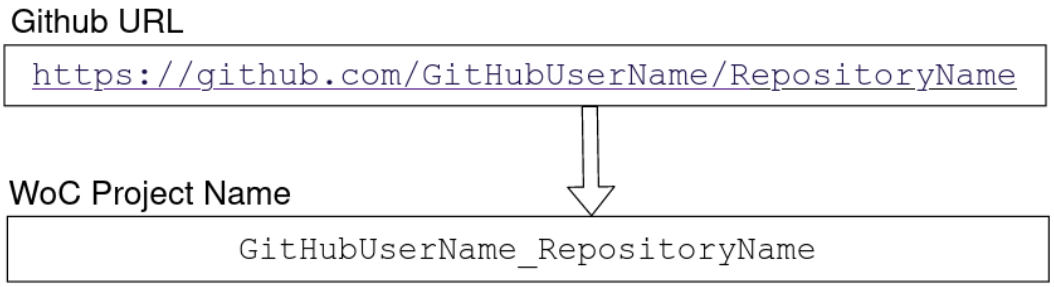


Figure 3. **Github URL to WoC project name translation**

3.3.2 Gathering the contents (blobs) of the project

The 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 I 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, I collected 1,659,435 commits generated in the hackathon repository. Then, I 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.

```
$ cat PrList | ~/lookup/getValues -f p2c > p2c.csv
$ cat p2c.csv | cut -d\; -f2 | sort -u > CsList
$ cat CsList | ~/lookup/getValues -f c2b > c2b.csv
```

3.3.3 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 the aim of this project is the identification of the reuse of “code", I decided to filter the blobs to only have the ones related to “code". In order to achieve that, I 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, I 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 I am focusing on in this study. Additionally, I 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, I 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.

```
$ cat c2b.csv | cut -d\; -f2 | sort -u > BsList
$ cat BsList | ~/lookup/getValues b2f > b2f.csv
```

3.3.4 Gathering data required to identify the origins of hackathon code (RQ1)

To address the first research question, I 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. I extracted the *author* of that first commit, along with the *timestamp*,

which would be useful in identifying when the blob was first created. The WORLD OF CODE provides the timestamps in Linux format, so I converted the timestamps from Linux based timestamp to a readable date/time format by using the awk[7] tool in Linux.

```
$ cat BsList | ~/lookup/getValues b2a > b2aLTS.csv
$ cat b2aLTS.csv | awk -F\; '{OFS=";"; $2=strftime("%Y-%m-%d %H:%M:%S
    ", $2); print $0}' > b2a.csv
```

I 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. I considered 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 [46, 7]. I 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 the assumption would be valid for most of the hackathons under consideration. Under this assumption, I have the start and end dates of each of the hackathon events, and I used that information to identify if a blob used in a hackathon project was created before, during, or after the hackathon event.

Furthermore, In order to analyze from the developers' perspective, I can identify all the developers who committed to the hackathon project using the project-to-author map (*p2a*) in WORLD OF CODE. I can also 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 [8] using WORLD OF CODE. With this data, I 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, I used WORLD OF CODE to identify the project associated with the first commit for each blob under consideration using

[7] `https://linux.die.net/man/1/awk`

[8] I used the approach outlined by [21] for author ID disambiguation to merge all of the different IDs belonging to one developer together, which is a common occurrence, as discussed in [10]

commit-to-Project map (*c2P*) in WORLD OF CODE, identified all developers of that project using project-to-author map (*p2a*) in WORLD OF CODE, 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 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).

```
$ cat PrList | ~/lookup/getValues p2a > p2a.csv
$ cat b2a.csv | cut -d\; -f4 | sort -u > CsFirstList
$ cat CsFirstList | ~/lookup/getValues -f c2P > c2PFirst.csv
$ cat c2PFirst.csv | cut -d\; -f2 | sort -u > PsFirstList
$ cat PsFirstList | ~/lookup/getValues p2a > p2aFirst.csv
```

3.3.5 Gathering data to identify hackathon code reuse (RQ2)

The second research question focuses on the reuse of *hackathon code*, which, per the definition (see section 1), refers to the blobs created during the hackathon event by one of the members of the hackathon team. Therefore, to address this question, I utilized the results of the 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, I collected all commits containing these blobs from WORLD OF CODE using the blob-to-commit (*b2c*) map, and I 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 with each

commit, excluding the forked ones (based on the work published in [40]), and I 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.

```
$ cat BsList | ~/lookup/getValues b2c > b2c.csv
$ cat b2c.csv | cut -d\; -f2 | sort -u > CsAllList
$ cat CsAllList | ~/lookup/getValues -f c2P > c2PAll.csv
```

3.3.6 Gathering hackathon related characteristics that affect code reuse (RQ3)

In addition to tracking hackathon code reuse (RQ2), I also aimed to study factors that can affect this phenomenon. For this purpose, I 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 1.1.

For the variables related to **H1**, The familiarity of the project, I selected the number of participants. I also selected if the hackathon event is co-located as an indication of familiarity. Both these attributes are rather straightforward since both exist in DEVPOST dataset so they were used directly.

For the variables related to **H2**, project Prolificness, I selected the number of technologies attribute as an input for this hypothesis, which exists in DEVPOST. I also selected the number of blobs in the hackathon project that was originally created before and during the event as an input. These two attributes are calculated based on the data collected for RQ1 from WORLD OF CODE and the analysis conducted for RQ1 in section 3.4.1.

For the variables related to **H3**, the composition of the repository, I 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. I 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, I 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 2.

Table 2. **Description of variables used for addressing RQ3. For the Binary variable, no. of TRUE/FALSE cases are shown**

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

The data I collected for RQ2 was for the blobs, so, in order to find out if code from a project were reused, I 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, I 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 the analysis, I excluded the hackathon projects with a single member, since a hackathon project "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, I also automatically filtered out any project that had no code blobs in its repository. After applying all the filtering, I was left with 12,881 hackathon projects.

3.4 Analysis procedure

3.4.1 Analysis Method to identify the origins of hackathon code (RQ1)

The first research question focus on how the code gets generated in the hackathon event, do the teams use their own code, did they reuse existing code they have, or do they use code that doesn't relate to them. All the collected data in section 3.3.4 are joined together using pandas built-in Merge function with different parameters like left, right, and inner joins to merge the datasets. Later I continued transforming the merged datasets to generate one final dataset ready for analysis. The attributes and their description is explained in table 3. The dataset contains information about the hackathon project itself like devpostID, ProjectID, Project Author list (PAuthorL), hackathon start/end dates, BlobHash, Blob timestamp, Blob First Commit Hash, Blob Author, and Commit Author List.

Table 3. List of Attributes of code generation dataset

Dataset Name	List of Attributes	Meaning
Code Generation Dataset	DevpostID	DEVPOST dataset primary key
	ProjectID	Hackathon Project ID
	HAuthorL	Hackathon project author list
	hackathonStartDate	Hackathon event Start Date
	hackathonEndDate	Hackathon event End Date
	BlobHash	Hackathon Project Blobs
	BTimeStamp	Hackathon Blob First usage timestamp
	BCommitHash	Hackathon Blob First usage Commit Hash
	BAuthor	Hackathon Blob first author
	PAuthorL	First Commit project authors List
	FirstAuthorL	First Author including other alias (if exist) List

I developed two flags which are calculated in this phase, the first one is called

TimingFlag which indicates if the blob was originally created before, during or after the hackathon event. The second flag is named ***AuthorFlag*** which indicates if the blob was originally created by one of the members of the hackathon project (ProjectMember) 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 (OtherAuthor). Table 4 summarize the values for all the classification flags.

Table 4. Classification flags for RQ1 with values definition

Flag Name	Values	Meaning
TimingFlag	1	Blob is created before the hackathon
	2	Blob is created during the hackathon
	3	Blob is created after the hackathon
AuthorFlag	1	Blob author is part of the Hackathon team author
	2	Blob author is sharing contribution with hackathon team
	3	Blob author is not related to hackathon team

I started building the logic for calculating the TimingFlag based on the dataset in table 3. The logic for identifying the TimingFlag is relatively simple, I basically compare the first commit timestamp of each blob with the hackathon start/end date, based on that, I set the flag to one of the three possible values. The logic used to build the TimingFlag is shown in algorithm 1

Algorithm 1: TimingFlag Calculation Logic

Input: Dataset no. 1
Result: TimingFlag column is filled with correct value

```
1 foreach row in the dataset do
2     if hackathonStartDate <= BTimeStamp <= hackathonEndDate then
3         return 2;
4     else if BTimeStamp > hackathonEndDate then
5         return 3;
6     else
7         return 1;
```

The logic for calculating the AuthorFlag mainly relies on identifying the original

author of each blob under consideration and comparing it with the hackathon team members. Due to the author ID disambiguation discussed in section 3.3.4, I am also comparing not only the author name, but also the other aliases available in the attribute FirstAuthorL. I built python logic to generate the flag based on algorithm 2.

Algorithm 2: AuthorFlag Calculation Logic

Input: Dataset no. 1
Result: AuthorFlag column is filled with correct value

```
1 foreach row in the dataset do
2     if list(set(FirstAuthorL) & set(HAuthorL)) > 0 then
3         return 1;
4     else if list(set(HAuthorL) & set(PAuthorL)) > 0 then
5         return 2;
6     else
7         return 3;
```

3.4.2 Analysis Method to identify hackathon code reuse (RQ2)

The second research question focuses on hackathon code reuse and which open-source projects reused the hackathon code. The dataset in table 5 was created to serve and answer the second research question. The dataset contains some attributes from the previous analysis (TimingFlag and AuthorFlag) plus some more information. TimingFlag represents when the hackathon code was first created, it can have three values that indicate if the code was created before, during, or after the hackathon. The second flag is AuthorFlag, which represents who created that blob, which is also based on the analysis of the first research question. The AuthorFlag can have three values which indicate if the blob is created by one of the hackathon team member *(SameAuthor)*, created by other developers who at some point was part of a project where one of the hackathon team members participated *(CoAuthors)* or created by another developer who is not related to the hackathon team. The dataset also contains information about other projects that used the hackathon blobs like *CProject* and *PauthorL*. The list of attributes and the

description of each one is clarified in table 5. The dataset with all these attributes is ready for analyzing the reuse of the hackathon code.

Table 5. List of Attributes of the code reuse dataset

Dataset Name	List of Attributes	Meaning
Code Usage Dataset	DevpostID	DEVPOST dataset primary key
	ProjectID	Hackathon Project ID
	hackathonStartDate	Hackathon event Start Date
	hackathonEndDate	Hackathon event End Date
	TimingFlag	Flag for the creation time of the blob
	AuthorFlag	Flag for the creator of the blob
	BlobHash	Hackathon Project Blob
	CommitHash	commits that uses the hackathon blob
	CTimeStamp	Commit Timestamp
	CAuthor	Commit Author
	CProject	Commit Project
	PAuthorL	Commit Project Author List
	PAuthorCount	Commit Project Author count
	PSizeFlag	Commit Project Size (Small, Medium, Large)

In addition to understanding how the blobs get reused, I also wanted to understand if they are reused in very small projects, or if larger projects also reuse these blobs. So, I needed a way to classify the projects into different categories. I 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 developers and *stars* are quintessential measures of project size and popularity and were found in figure 7 to have a low correlation (Spearman Correlation: 0.26), so I decided to use both measures.

Instead of manually classifying the projects using these variables using arbitrary thresholds, I decided to use *Hartemink's pairwise mutual information-based discretization method* [25], 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. I 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”.

I then defined a new output variable ***(UsageFlag)*** which can have several values based on the target project that used the hackathon blob. The UsageFlag logic is based on checking the commit which used the hackathon blob and checked how large this project is. Based on that I assign a value that represents if this targeted project is the same hackathon project *(UsageFlag=1)*, the target project is a different project with small size *(UsageFlag=3)*, the target project is a different project with medium size *(UsageFLag=4)* or the target project is a different project with large size *(UsageFlag=5)*. Table 6 shows the summary of the values and the meaning of each one.

Table 6. Classification flags for RQ2 with values definition

Flag Name	Values	Meaning
UsageFlag	1	Blob is reused in the same hackathon project
	3	Blob is reused in a small project
	4	Blob is reused in a medium project
	5	Blob is reused in a large project

The analysis conducted in this step is on project-blob-commit level (Lowest level) since I am comparing each hackathon blob with every commit that used this blob afterward and checks the project size to identify the flag value. For instance, If one hackathon blob is used in five commits after the hackathon event, I will have five records in the dataset and the same hackathon blob will have five different values for the UsageFlag. In order to see the maximum size of projects this hackathon blob was used, I grouped the dataset based on the hackathon blobs and got the maximum of the UsageFlag. This technique allowed me to see to what extent the hackathon blob was used. For example, If a hackathon blob was used twice in small and medium projects, so the maximum value of UsageFLag will become 4 since the value for the medium-sized projects is greater than the value for small ones. Algorithm 3 shows the logic used to

build the UsageFlag.

Algorithm 3: UsageFlag Calculation Logic

Input: Dataset no. 2
Result: UsageFlag column is filled with correct value

```
1 foreach row in the dataset do
2     if ProjectID == CProject then
3         return 1;
4     else if row.PSizeFlag == 1 then
5         return 3;
6     else if row.PSizeFlag == 2 then
7         return 4;
8     else if row.PSizeFlag == 3 then
9         return 5;
```

3.4.3 Analysis Method for Identifying project characteristics that affect code reuse (RQ3)

As noted in the hypotheses presented in section 1.1, I am 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 the analysis is not to make the best predictive model that gives the optimum predictive accuracy, instead, I am trying to find out which of the predictors have a significant effect by creating an explanatory model. As noted by Shmueli [53], these two are very different tasks.

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

Since I presumed the variables related to **H1** and **H2** would have a linear effect on the probability of code reuse for a project, I 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 I 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)*

3.5 Case study analysis

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

From the analysis results in section 4.1 where I was able to identify the hackathon projects code when they were originally created. I created a dataset that includes each hackathon project and the count of blobs originally created before, during and after the hackathon event. I 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 afterward (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, I looked at both projects that had some of their code reused, and projects that did not, and chose some projects 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 the results section in table 8.

I rerun the same collection defined in section 3.3 for the selected projects from each category following the same data collection procedure defined in section 3.3 but without filtering the code blobs only, since the aim here is to study these projects and see the behavior of the blobs reused. I started by collecting the commits and blobs related to these projects using ***p2c*** and ***c2b*** maps in WORLD OF CODE respectively. I then collected

the original usage of each blob with a timestamp using the ***b2a*** map. I also collected the hackathon project authors using the ***p2a*** map. I then collected all the commits that use the hackathon blob using the ***b2c*** map and the projects that had those commits using the ***c2P*** map. I also collected the blob types from the Github Linguist tool described in section 3.3.3 without applying the filter in order to identify the file types.

Based on the collected data, I applied the same analysis procedure described in section 3.4 in order to identify the three flags *(TimingFlag, AuthorFlag, UsageFlag)*. At this point, I generated a dataset including all the hackathon projects under consideration with all the blobs included and the other projects that used these blobs after the hackathon event. I then conducted a manual investigation on the cases where usage was determined in order to get more insights about file types reused, if license files exist in the repository, is the reuse related to a framework or a library, etc..

4 Results

Here I will discuss the findings in relation to the research questions and discuss the result of a small case study on some examples of code reuse for selected hackathon projects. I will start by showing the analysis result for the origins of hackathon code *(RQ1)*, then I am showing the results for hackathon code reuse related to *(RQ2)*, followed by the results for the hackathon project characteristics affects code reuse *(RQ3)*. The last section here explain the results of the case study conducted on a small set of projects.

4.1 Origins of hackathon code

As mentioned in section 1.1 and section 3.3.4, I 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", I examined if the first creation of the code blob under consideration was *Before, During*, or *After* the corresponding hackathon event. In terms of "who", I 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

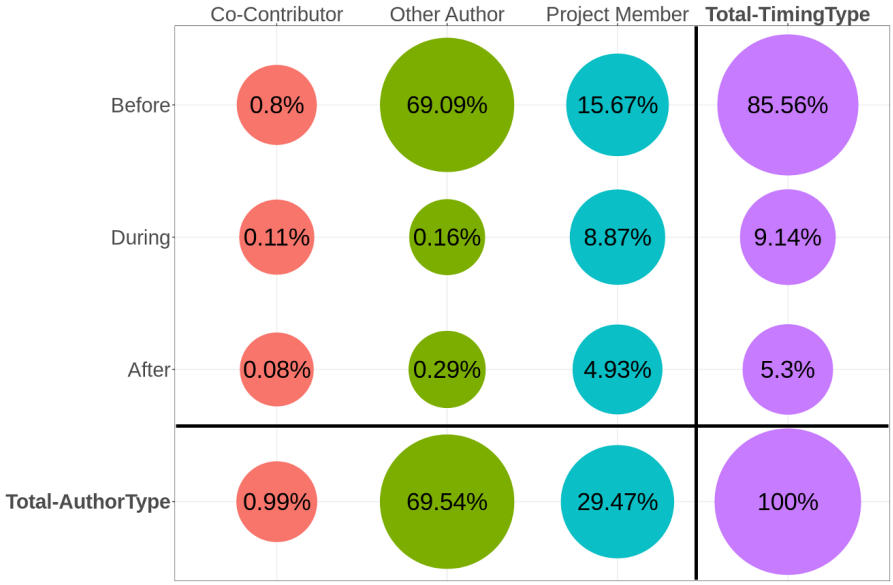


Figure 4. **Plot of Who created how much of the Hackathon Code and When**

the hackathon project contributed to as well (*Co-Contributor*), or someone else (*Other Author*).

The result of the analysis is presented in fig. 4, 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. Around 9.14% of the blobs were created during events and 5.3% of the blobs were created after an event.

Looking at top languages for the blobs created at different times (fig. 5), identified by the GITHUB *linguist* tool, I 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 I 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.

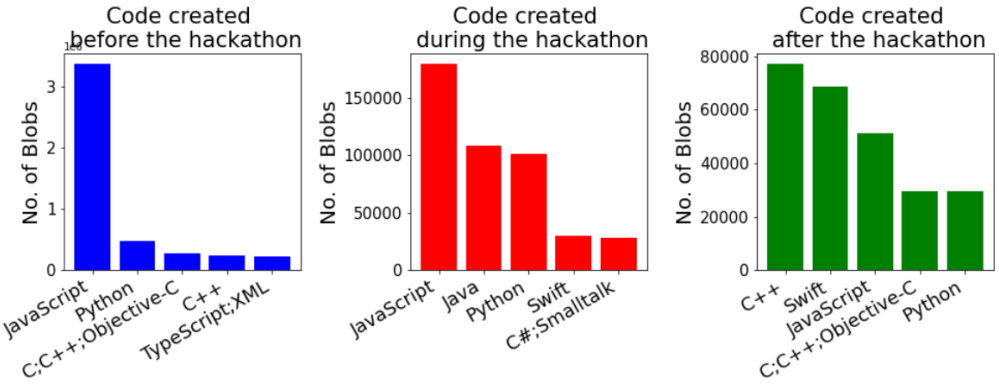


Figure 5. **Top 5 languages for blobs created before, during, and after hackathons**

Figure 4 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. 6, I 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. 5). This is not surprising, since they have a great deal of overlap (fig. 4). Most of the code created by project members indicates a leaning towards web/mobile app development, and most of the C++ and Python code was found to be related to Machine Learning frameworks.

4.2 Hackathon code reuse

As discussed in section 1.1 and section 3.3.5, the goal while looking for hackathon code reuse is twofold: First, I want to see how much of the code gets reused, and second, I want to find if they get reused in small, medium, or large projects. By following the procedure outlined in section 3.3.5, I found that 167,781 (28.8%) of the 581,579 hackathon code blobs got reused in other projects.

I further classified the projects that reused these code into *Small, Medium*, and *Large*, as discussed in section 3.4.2 based on the Number of Stars and the Number of Developers attributes since they have low correlation (Spearman Correlation: 0.26) as explained

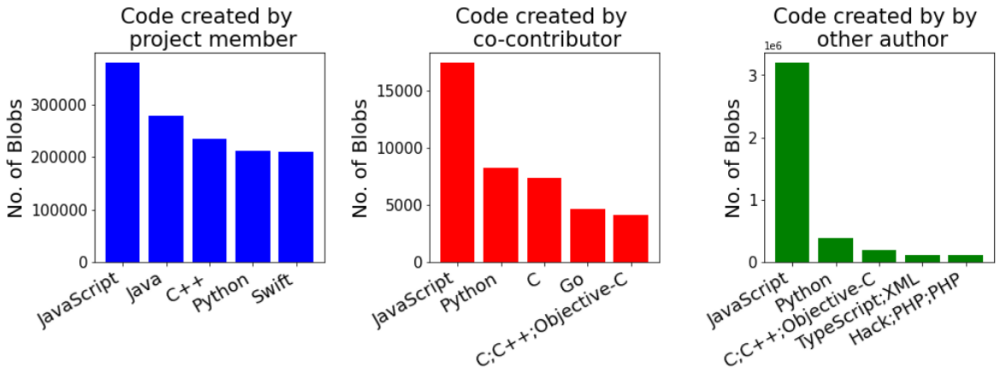


Figure 6. **Top 5 languages for blobs created by project members, co-contributors, and others**

in section 3.4.2. Overall, I identified 1,368,419 projects that reused at least one of the 581,579 blobs, and using the classification, 1,220,114 (89.2%) projects were classified as "Small", 116,177 (8.5%) as "Medium", and 32,128 (2.3%) as "Large". Figure 7 shows the correlation between all variables collected from WORLD OF CODE.

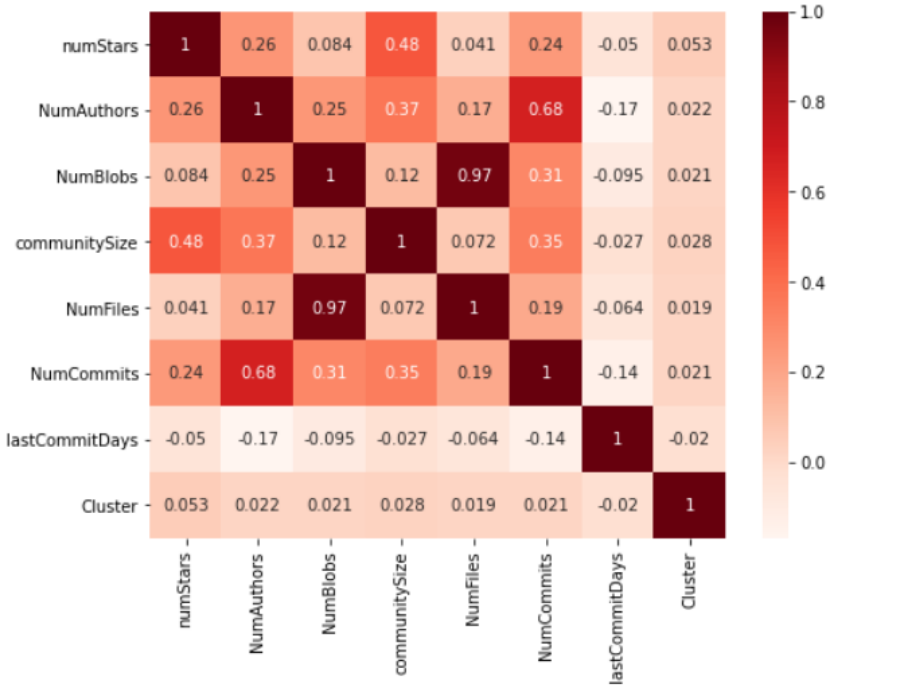


Figure 7. Spearman correlation Heatmap for project characteristics

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 I 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 figure 8, I 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. 9. I can

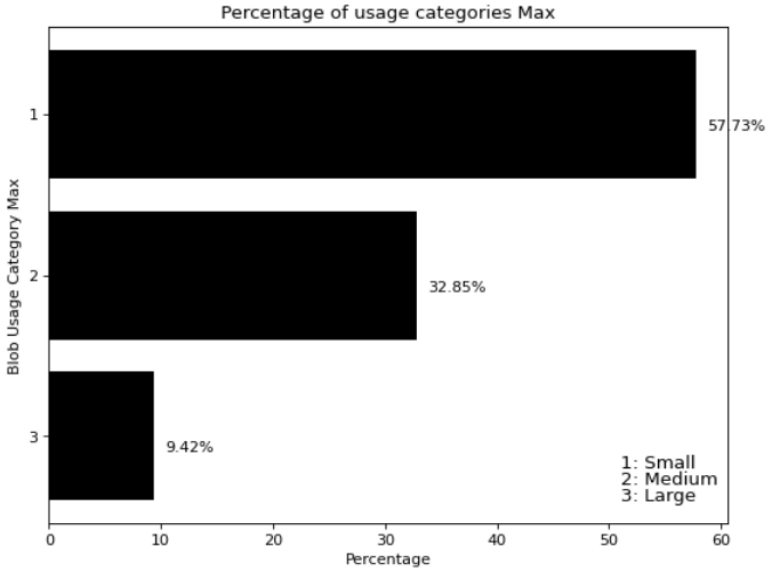


Figure 8. Plot depicting Hackathon code reuse in projects of different categories when only the maximum size is considered

see that 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, I 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.

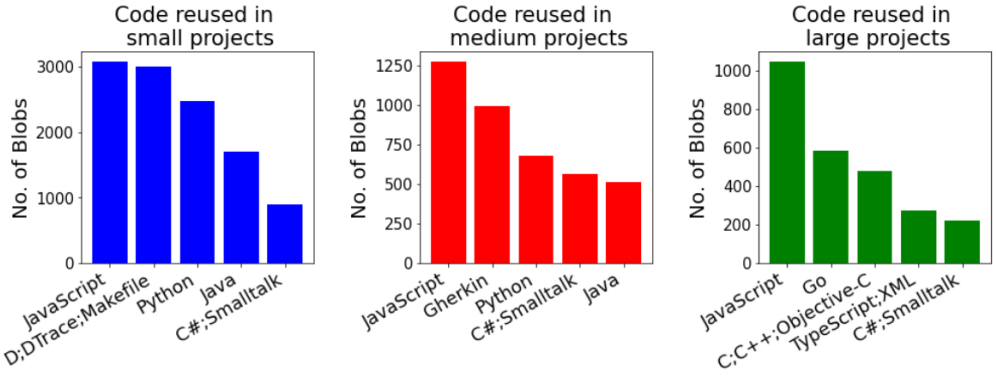


Figure 9. **Top 5 Languages for the reused code blocks in different projects**

I was interested in exploring the temporal dynamics of code reuse as well. Therefore, I 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. 10, which shows the *weekly* hackathon code reuse for 2 years after the end of the corresponding hackathon event, with the fraction of the total number of hackathon code blobs (581,579) reused per week on the Y-axis. Moreover, I can see from this plot that, 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.

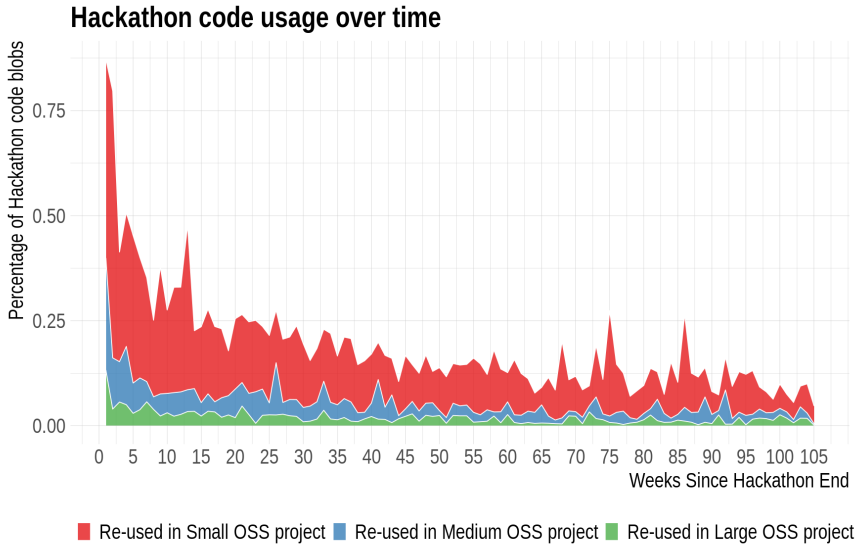


Figure 10. **Plot depicting hackathon weekly code reuse in projects of different categories over the period of 2 Years**

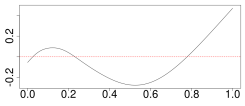
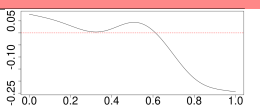
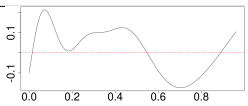
4.3 Characteristics affecting code reuse

The third and final Research Question was about identifying what project characteristics affect code reuse and I formed three hypotheses about what factors might be affecting it, which were presented in section 1.1. Using the procedure outlined in section 3.3.6, I gathered the variables of interest related to the three hypotheses, as presented in table 2.

As discussed in section 3.4.3, I decided to use Generalized Additive Models (GAM) to identify the variables that have a significant impact on code reuse.

The result of the analysis is presented in table 7, which shows that the number of participant, number of technology and if the event is colocated are having significant effect on code reuse. The effect direction, communicated by the signs of the estimates for the corresponding variables match the rationale I presented in section 1.1.

Table 7. Effect of Project Characteristics on Hackathon Code Reuse - Results from the Generalized Additive Model.
Part A. showing the results for the *linear* terms, with the associated Estimate, Standard Error, and p-Values.
Part B. shows the results for the *non-linear* terms, with the Effective Degrees of Freedom – "edf" – a measure of the degree of non-linearity, the p-Values, and the partial effects of each variable on the response (0: No Effect, Positive Values: Positive effects, Negative Values: Negative Effects).
The "pctData" variable, found to be "not significant", is shown in RED, and the corresponding effect plot is omitted

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	

As for the variables related to **H3**, the "pctData" (see table 2 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, I decided to observe their partial effect plots, which show the relationship between the two plotted variables (an outcome and an explanatory variable) while adjusting for interference from other explanatory variables [60]. 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.

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 I observed for the "pctProse" variable.

4.4 A Case-Study on Code Reuse

In addition to investigating the research questions presented here, I also conducted a small-scale case study on a few projects selected by stratified random sampling to gain additional insights. Since, for example, I know from the earlier analysis that some of the code is reused, but I do not always know if these are part of some framework, generated by some scripts, or if there are other file types that also get reused.

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 the results section in table 8. The 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.

• *tudorv91_SparkJNI:* This project has a README and license files, The project is cloned to another repository where all hackathon blobs are reused there. No other reuse occurred on these blobs.

• *TheMichaelHu_PickyPusheen*: A number of reuses, mostly icons and configuration

Table 8. 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
	tudorv91_SparkJNI *(Type-A)*	297	41	*1473*	Reused	24	3	0	13	1
					Not reused	0	0	0	0	0
	TheMichaelHu_PickyPusheen *(Type-B)*	*3372*	520	1	Reused	66	3	0	9	1
					Not reused	90	259	2	78	12
	IsaiahJTurner_Contap *(Type-B)*	*2106*	122	0	Reused	2	6	0	22	1
					Not reused	40	22	0	29	0
	smit-happens_childs-play *(Type-B)*	513	162	0	Reused	7	1	0	1	1
					Not reused	104	24	0	18	6
	drfuzzyness_WearHax-OlivMatt *(Type-C)*	3154	823	1194	Reused	4	60	0	17	2
					Not reused	19	645	0	53	4
	Opportunity-Hack-2016-AZ_Team18 *(Type-C)*	71	59	180	Reused	0	0	0	1	0
					Not reused	3	4	22	28	1
	kylemsguy_building-point *(Type-D)*	29	*81*	10	Reused	7	7	0	15	0
					Not reused	0	26	0	26	0
	davidemily_tigerhack2017 *(Type-D)*	22	*119*	0	Reused	3	1	43	52	20
					Not reused	0	0	0	0	0
	itcarrillo_GifMe *(Type-D)*	5	*131*	6	Reused	1	12	51	65	2
					Not reused	0	0	0	0	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
	AmirBraham_TouchWizards *(Type-A)*	2855	155	*6403*	Used	1	0	0	0	0
					Not reused	42	88	0	24	0
	cnorick_randomRestaurant *(Type-A)*	2	5	*30*	Used	0	0	0	0	0
					Not reused	0	1	0	4	0
	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
	ProjectProgramAMark_fluffy-journey *(Type-C)*	643	44	50	Reused	0	1	1	0	0
					Not reused	0	6	13	22	1
	Hackception_global-hack-6 *(Type-C)*	37	213	46	Reused	0	1	0	0	1
					Not reused	0	44	94	64	9
	jrdbnntt_DaReactTV *(Type-D)*	4	*63*	1	Reused	2	0	0	0	0
					Not reused	2	11	6	41	1
	liujordan_TestYourTech *(Type-D)*	13	*160*	0	Reused	0	1	0	0	0
					Not reused	0	4	33	115	7
	cereal_fishystocks *(Type-D)*	120	*147*	92	Reused	0	1	0	0	0
					Not reused	40	10	7	87	2

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.

• *IsaiahJTurner_Contap:* This project does not have a README file nor a license file. 21 reused code blobs created during the hackathon are created by someone else who is not part of the hackathon project team, one reused code blob is first time created by one of the hackathon team members, The file is a java script file under node modules, the file is used in 3 different projects.

• *smit-happens_childs-play:* One code blob created by one of the hackathon team members is reused 121 times in 103 projects. file *3DTextShader.shader* is part of assets folder, used for 3D Text shader.

• *Opportunity-Hack-2016-AZ_Team18:* This project has only one file *(validator_tool.js)* get reused in one other project and the file was added by the same developer who participated in the hackathon.

• *davidemily_tigerhack2017:* This project doesn't exist anymore on GITHUB, however, it's blobs are reused in two different projects with names *jpk2f2_tigerhack2017* and *davidemily_TigerNews* which may indicate that the same user cloned the repository to another name.

• *smit-happens_childs-play:* This project shows evidence of code reuse related to Microsoft HoloLens which was widely reused afterwards by projects of different sizes. The file types used are exe and dll files with some files related to Unity game engine. This project has a README with less documentation.

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. There are instances observed where the full hackathon project is cloned to another project especially in Type *D* which may indicate that some developers copy their code across multiple projects. It is also worth noting that for most of the code reuses I observed, the code was part of a package/library/framework used in the project. This is not too surprising, since I am only looking at *exact* code reuse. However, I filtered the 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.

5 Discussion

In this section, I would like to dive deeply into the collected results and discuss their relationship to our research questions and existing literature. I am also discussing about the limitations and the implications on research and practice.

5.1 Answering the research questions

5.1.1 Origins of hackathon Code RQ1

The result of the analysis related to Hackathon Code Generation RQ1 which is presented in section 4.1, 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 [39].

The figure 4 shows that around 9.14% of the blobs were created during events, since participants need to be efficient during an event owing to the time limit [47] which fosters reuse as previously discussed in the context of OSS [24]. I 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 [44].

> ***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).*

If I consider the code blobs in created during and after the hackathon event, as shown by the combined picture in fig. 4, which are the main contributions of the hackathons in terms of code creation, I 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, I also find that the project members often reuse the code they had written earlier in their projects, 15.67% of all the code belongs 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 [44] and/or making (detailed) plans on what they want to achieve during an event [47].

> ***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).*

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 the assumption) by teams of around 2-3 members (up to a maximum of 10 members — see table 2), is indeed significant and it highlights the importance of hackathons in generating new code.

Therefore, to give a more complete answer to RQ1:

> ***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.*

5.1.2 Hackathon code reuse RQ2

By looking into the graph in figure 10, 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. [44] 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 continues 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 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.

From both graphs in figure 8 and 10, we can conclude the following answer to RQ2:

> ***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.*

5.1.3 Characteristics affecting code reuse RQ3

The result of the analysis is presented in table 7, which shows that all of the variables related to hypotheses **H1** and **H2**, which I 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 I presented in section 1.1. 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 increased continuation activity [44]. In that study, the team size was however negatively related to project continuation while for the study in this paper I 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 [47].

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.), I 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, I see that projects with almost all of their content as documentation are more likely to have their code reused. On closer inspection, I 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. [45, 58, 5]). the 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", I 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 I 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.

From both graphs in table 7 , we can conclude the following answer to RQ3:

Characteristics affecting Code Reuse (RQ3): *The hypotheses presented in section 1.1 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*** *was more complex, which led to additional insights about hackathon code reuse.*

5.2 Implications for practice

The findings have a number of implications for research and practice. They can serve as 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 also spend time on not only developing code but also providing additional materials and documentation, and also for the teams to reuse existing code 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.

5.3 Implications for research

With respect to research, the findings provide an initial account of how code gets reused and created during a hackathon as well as whether and where it gets reused afterward. 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 the 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.

There are several ways to extend this research, One interesting aspect is to consider code clones/snippets while looking for code reuse (e.g. by looking at the associated CTAG tokens - a dataset available in WORLD OF CODE). It may be common that developers copy code files and do changes while developing, this will generate a new blob hash and the current approach I followed doesn't consider that as reuse.

In the current study, I considered each hackathon project as a standalone entity, and I didn't track code usage and spread between different hackathons and hackathon projects. However, some instances of project clone were observed while doing the case study and it can be common that developers participate in different hackathons and utilize the same

code they developed earlier in each hackathon project. So it may worth studying how the hackathon code propagates between different hackathons.

Other research directions can be to identify other factors that affect code reuse, include code quality [13, 14], project popularity [12, 11], 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 [15, 16] which might aid in the understanding of hackathon code reuse as well.

I 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.

5.4 Limitations and Threats to Validity

For the study, I 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 I focused only on exact code reuse since any changes in the file contents would lead to a change in the blob SHA1 value that I 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 the study, nor can I capture the reuse of code snippets. Moreover, the analysis does not consider the size of a file since I am looking at the SHA1 values of the blobs, i.e. the analysis cannot distinguish between the 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 the research questions. I 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 I studied which may affect the results of RQ1 and RQ3.

I relied on the GITHUB Linguist tool to categorize files and I only focused on files with the type "Programming", however, 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 the study I only considered hackathon projects, thus, the findings may not be generalizable to other types of software projects and repositories.

6 Conclusion

In this study, I investigated the origins of hackathon code and its reuse after an event. I found that most hackathon projects reuse existing code and that code created during events also gets reused by other OSS projects later on. The 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, the 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.

Acknowledgements

First and foremost, I would like to express the sincerest gratitude to my supervisor, Assoc. Prof. **Alexander Nolte**, for their guidance and support during my master's research. Despite the challenging times, his encouragement, critical feedback, guidance and support led me to complete my masters thesis with grace

The work presented here was accepted and published in the hackathon paper track in The Mining Software Repositories (MSR) conference 2021 under the name "Tracking hackathon code generation and reuse" which is co-authored by **Tapajit Dey**[9] during MSR hackathon in December 2020. The preprint of the paper is available in [29].

Moreover, Based on the results presented here, another paper was written and accepted in the technical paper track in Mining Software Repositories (MSR) conference 2021 under the name "The Secret Life of Hackathon Code" which is co-authored by **Tapajit Dey**[9], **Alexander Nolte**[10,11], **Audris Mockus**[12], and **James D. Herbsleb**[11]. Thus, I would like to thank them all for the help and the precious time I spent with them during the hackathon and while writing the technical paper. The preprint of the paper is available in [30].

Ahmed Samir Imam Mahmoud

May 2021

[9] Lero—the Irish Software Research Centre, University of Limerick, Limerick, Ireland
[10]University of Tartu, Estonia
[11]Carnegie Mellon University, Pittsburgh, PA, USA
[12]University of Tennessee, Knoxville, TN, USA

References

[1] Replication package. https://github.com/woc-hack/track_hack.

[2] World of code tutorial. https://github.com/woc-hack/tutorial.

[3] Rabe Abdalkareem, Emad Shihab, and Juergen Rilling. On code reuse from stackoverflow: An exploratory study on android apps. *Information and Software Technology*, 88:148–158, 2017.

[4] Bastiaan Baccarne, Peter Mechant, Dimitri Schuurma, Lieven De Marez, and Pieter Colpaert. Urban socio-technical innovations with and by citizens. *Interdisciplinary Studies Journal*, 3(4):143, 2014.

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

[6] Aaron Ciaghi, Tatenda Chatikobo, Lorenzo Dalvit, Darsha Indrajith, Mfundiso Miya, Pietro Benedetto Molini, and Adolfo Villafiorita. Hacking for southern africa: Collaborative development of hyperlocal services for marginalised communities. In *IST-Africa Week Conference, 2016*, pages 1–9. IEEE, 2016.

[7] David Cobham, Kevin Jacques, Carl Gowan, Jack Laurel, Scott 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.

[8] R Cameron Craddock, Daniel S Margulies, Pierre Bellec, B Nolan Nichols, Sarael Alcauter, Fernando A Barrios, Yves Burnod, Christopher J Cannistraci, Julien Cohen-Adad, Benjamin De Leener, et al. Brainhack: a collaborative workshop for the open neuroscience community. *GigaScience*, 5(1):16, 2016.

[9] Karen Day, Gayle Humphrey, and Sophie Cockcroft. How do the design features of health hackathons contribute to participatory medicine? 2017.

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

[11] Tapajit Dey, Yuxing Ma, and Audris 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*, PROMISE'19, page 36–45, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Tapajit Dey and Audris 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*, PROMISE'18, page 66–69, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Tapajit Dey and Audris 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*, PROMISE'18, page 56–65, New York, NY, USA, 2018. Association for Computing Machinery.

[14] Tapajit Dey and Audris Mockus. Deriving a usage-independent software quality metric. *Empirical Software Engineering*, 25(2):1596–1641, Mar 2020.

[15] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. Detecting and characterizing bots that commit code. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 209–219, New York, NY, USA, 2020. Association for Computing Machinery.

[16] Tapajit Dey, Bogdan Vasilescu, and Audris Mockus. An exploratory study of bot commits. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 61–65, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Jeanette Falk Olesen and Kim Halskov. 10 years of research with and on hackathons. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, pages 1073–1088, 2020.

[18] Daniel Feitosa, Apostolos Ampatzoglou, Antonios Gkortzis, Stamatia Bibi, and Alexander Chatzigeorgiou. Code reuse in practice: Benefiting or harming technical debt. *Journal of Systems and Software*, page 110618, 2020.

[19] Anna Filippova, Erik Trainer, and James 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*, pages 152–163. IEEE Press, 2017.

[20] Allan 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*, pages 38–41. ACM, 2016.

[21] Tanner Fry, Tapajit Dey, Andrey Karnauch, and Audris 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*, MSR '20, page 518–522, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Kiev Gama, Breno Alencar, Filipe Calegario, André Neves, and Pedro Alessio. A hackathon methodology for undergraduate course projects. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2018.

[23] Daniel 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)*, pages 24–24. IEEE, 2007.

[24] Stefan Haefliger, Georg Von Krogh, and Sebastian Spaeth. Code reuse in open source software. *Management science*, 54(1):180–193, 2008.

[25] Alexander John Hartemink. *Principled computational methods for the validation discovery of genetic regulatory networks*. PhD thesis, Massachusetts Institute of Technology, 2001.

[26] Alexis Hope, Catherine D'Ignazio, Josephine Hoy, Rebecca Michelson, Jennifer Roberts, Kate Krontiris, and Ethan Zuckerman. Hackathons as participatory design: Iterating feminist utopias. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 61. ACM, 2019.

[27] Youyang Hou and Dakuo Wang. Hacking with npos: collaborative analytics and broker roles in civic data hackathons. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):1–16, 2017.

[28] Daniela Huppenkothen, Anthony Arendt, David W Hogg, Karthik Ram, Jacob T VanderPlas, and Ariel Rokem. Hack weeks as a model for data science education and collaboration. *Proceedings of the National Academy of Sciences*, 115(36):8872–8877, 2018.

[29] Ahmed Imam and Tapajit Dey. Tracking hackathon code creation and reuse. *arXiv preprint arXiv:2103.10167*, 2021.

[30] Ahmed Imam, Tapajit Dey, Alexander Nolte, Audris Mockus, and James D Herbsleb. The secret life of hackathon code. *arXiv preprint arXiv:2103.01145*, 2021.

[31] Naohiro Kawamitsu, Takashi Ishio, Tetsuya Kanda, Raula Gaikovina Kula, Coen De Roover, and Katsuro 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*, pages 305–314. IEEE, 2014.

[32] Hanna Kienzler and Carolyn Fontanesi. Learning through inquiry: A global health hackathon. *Teaching in Higher Education*, 22(2):129–142, 2017.

[33] Marko Komssi, Danielle Pichlis, Mikko Raatikainen, Klas Kindström, and Janne Järvinen. What are hackathons for? *IEEE Software*, 32(5):60–67, 2015.

[34] Hilmar Lapp, Sendu Bala, James P Balhoff, Amy Bouck, Naohisa Goto, Mark Holder, Richard Holland, Alisha Holloway, Toshiaki Katayama, Paul O Lewis, et al. The 2006 nescent phyloinformatics hackathon: a field report. *Evolutionary Bioinformatics Online*, 3:287, 2007.

[35] 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)*, pages 143–154, Los Alamitos, CA, USA, may 2019. IEEE Computer Society.

[36] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris 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)*, pages 143–154. IEEE, 2019.

[37] Yuxing Ma, Tapajit Dey, Chris Bogart, Sadika Amreen, Marat Valiev, Adam Tutko, David Kennard, Russell Zaretzki, and Audris 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.

[38] Maria Angelica Medina Angarita and Alexander 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.

[39] Audris 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)*, pages 7–7. IEEE, 2007.

[40] Audris Mockus, Diomidis Spinellis, Zoe Kotti, and Gabriel John 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*, MSR '20, page 513–517, New York, NY, USA, 2020. Association for Computing Machinery.

[41] Steffen Möller, Enis Afgan, Michael Banck, Raoul JP Bonnal, Timothy Booth, John Chilton, Peter JA Cock, Markus Gumbel, Nomi Harris, Richard Holland, et al. Community-driven development for computational biology at sprints, hackathons and codefests. *BMC bioinformatics*, 15(14):S7, 2014.

[42] Arnab Nandi and Meris Mandernach. Hackathons as an informal learning platform. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 346–351. ACM, 2016.

[43] Alexander 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*, pages 31–36, 2019.

[44] Alexander Nolte, Irene-Angelica Chounta, and James D Herbsleb. What happens to all these hackathon projects? - identifying factors to promote hackathon project continuation. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–26, 2020.

[45] Alexander Nolte, Linda Bailey Hayden, and James D Herbsleb. How to support newcomers in scientific hackathons-an action research study on expert mentoring. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–23, 2020.

[46] Alexander Nolte, Ei Pa Pa Pe-Than, Abasi-Amefon O. Affia, Chalalai Chaihirunkarn, A. Filippova, Arun Kalyanasundaram, Maria Angelica Medina Angarita, Erik H. Trainer, and James D. Herbsleb. How to organize a hackathon - a planning kit. *ArXiv*, abs/2008.08025, 2020.

[47] Alexander Nolte, Ei Pa Pa Pe-Than, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. You hacked and now what? -exploring outcomes of a corporate hackathon. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–23, 2018.

[48] Lavinia Paganini and Kiev 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*, pages 8–15, 2020.

[49] Ei Pa Pa Pe-Than and James D Herbsleb. Understanding hackathons for science: Collaboration, affordances, and outcomes. In *International Conference on Information*, pages 27–37. Springer, 2019.

[50] Ei Pa Pa Pe-Than, Alexander Nolte, Anna Filippova, Christian Bird, Steve Scallen, and James D Herbsleb. Designing corporate hackathons with a purpose: The future of software development. *IEEE Software*, 36(1):15–22, 2019.

[51] Jari Porras, Antti Knutas, Jouni Ikonen, Ari Happonen, Jayden Khakurel, and Antti Herala. Code camps and hackathons in education-literature review and lessons learned. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[52] Bard Rosell, Shiven Kumar, and John Shepherd. Unleashing innovation through internal hackathons. In *Innovations in Technology Conference (InnoTek), 2014 IEEE*, pages 1–8. IEEE, 2014.

[53] Galit Shmueli et al. To explain or to predict? *Statistical science*, 25(3):289–310, 2010.

[54] Manuel Sojer and Joachim Henkel. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of the Association for Information Systems*, 11(12):868–901, 2010.

[55] Arlin Stoltzfus, Michael Rosenberg, Hilmar Lapp, Aidan Budd, Karen Cranston, Enrico Pontelli, Shann Oliver, and Rutger A Vos. Community and code: Nine lessons from nine nescent hackathons. *F1000Research*, 6, 2017.

[56] Nick Taylor and Loraine Clarke. Everybody's hacking: Participation and the mainstreaming of hackathons. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 172. ACM, 2018.

[57] Nick Taylor, Loraine Clarke, Martin Skelly, and Sara Nevay. Strategies for engaging communities in creating physical civic technologies. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 507. ACM, 2018.

[58] Erik H Trainer, Arun Kalyanasundaram, Chalalai Chaihirunkarn, and James 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*, pages 1118–1130. ACM, 2016.

[59] Georg von Krogh, Sebastian Spaeth, and Stefan 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*, pages 198b–198b. IEEE, 2005.

[60] Miles D. Williams. peplot() for plotting partial effects. `https://rpubs.com/milesdwilliams15/328471`. Accessed: 2021-01-11.

[61] Bowen Xu, Le An, Ferdian Thung, Foutse Khomh, and David Lo. Why reinventing the wheels? an empirical study on library reuse and re-implementation. *Empirical Software Engineering*, 25(1):755–789, 2020.

Appendix

I. Glossary

World of Code (WOC) : This is a platform that collects all open source projects and perform mining to find relations related to Project, Author, Commits and Blobs.

Blob: Binary large object, it is a binary representation of a file.

SHA-1 Hashing: Hashing is a technique to generate a fixed length string of the input data, it is used in WORLD OF CODE to generate hashed strings for Commits and Blobs.

DEVPOST: DEVPOST is a popular hackathon database that contains data about hackathons including hackathon locations, dates, prizes and information about teams and their projects including the project's GITHUB repositories.

FLOSS: Free/Libre Open Source Software.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Ahmed Samir Imam Mahmoud**,

*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   The secret life of Hackathon code,

   *(*title of thesis)

   supervised by Alexander Nolte.

   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Ahmed Samir Imam Mahmoud

14/05/2021