The State of Practice on Virtual Reality (VR) Applications: an Exploratory Study on Github and Stack Overflow

Naoures Ghrairi, Sègla Kpodjedo
ETS Montreal
Montreal, Canada
{naoures.ghrairi.1, segla.kpodjedo}@etsmtl.ca

Amine Barrak, Fábio Petrillo, Foutse Khomh
Polytechnique Montreal
Montreal, Canada
{amine.barrak, fabio.petrillo, foutse.khomh}@polymtl.ca

***Abstract*—Virtual Reality (VR) is a computer technology that holds the promise of revolutionizing the way we live. The release in 2016 of new-generation headsets from Facebook-owned Oculus and HTC has renewed the interest in that technology. Thousands of VR applications have been developed over the past years, but most software developers lack formal training on this technology. In this paper, we propose descriptive information on the state of practice of VR applications' development to understand the level of maturity of this new technology from the perspective of Software Engineering (SE). To do so, we focused on the analysis of 320 VR open source projects from Github to determine which are the most popular languages and engines used in VR projects, and evaluate the quality of the projects from a software metric perspective. To get further insights on VR development, we also manually analyzed nearly 300 questions from Stack Overflow. Our results show that (1) VR projects on GitHub are currently mostly small to medium projects, and (2) the most popular languages are JavaScript and C#. Unity is the most used game engine during VR development and the most discussed topic on Stack Overflow. Overall, our exploratory study is one of the very first of its kind for VR projects and provides material that is hopefully a starting point for further research on challenges and opportunities for VR software development.**

***Index Terms*—Virtual Reality, state of practice, Software Quality, empirical study**

I. Introduction

> *"The good news is that virtual reality is here. The bad news is that something is still missing."*
> — Mychilo Stephenson Cline —

Virtual reality (VR) is a new medium that allows users to experience fully immersive experiences. It provides users with the ability to be transported to other places virtually, but with realistic sensations, paving the way for previously unimagined interaction and communication means. For a long time, virtual reality was out of reach for the average consumer. Thanks to recent advances by Google, Facebook-owned Oculus, and HTC, this is no more the case. Virtual reality is now present in many aspects of our life, and experts are predicting that its impact will be similar to the introduction of television, the Internet, or the smart phone.

With the commercial release in 2016 of major pioneering products such as the Oculus Rift and the HTC Vive, development of VR applications presents an interesting case of a possibly disruptive technology going from a few developers toying with Software Development Kits (SDKs) to a growing number of technology start-ups and independent developers. VR has been proven to be a potential boon for small development teams: e.g., the game Holoball (a sort of VR Pong) admittedly developed by a small team of two developers in a 2-3 weeks period of time was a best seller on Steam (around 30k sales)[1]. Given the relatively low maturity of the field and the slow (but steady) pace of involvement from the big game studios, opportunities abound for developers and standards and best practices are still to be observed and set.

This reality is not lost on some of the gaming industry heavy-weights as they make aggressive moves to position their products as the top choice for VR development. We are witnessing a rapid growth of the number of software that can create and display consumer virtual reality. Frameworks like Unity3D and Unreal game engines, which are popular tools for desktop and mobile games developments, are also becoming the tools of choice for VR developers. On the web side, the powerful features offered by WebGL and 3D JavaScript frameworks (e.g., Three.js) are boosting the development of browser-based virtual reality applications.

These developments are worth analyzing and studying and this is what this paper sets out to do: as VR is about to go mainstream, what can we observe from a software engineering perspective: teams, releases, languages, engines, quality. Particularly interesting in this context is open source material, whose pioneering effect and standard setting potential is certainly greater than in most mature industries, especially given the mostly small companies that are trying to make a name in the game. In other words, we believe that, since VR is a nascent industry, open source work will be quite influential for (and reused by) a great number of developers trying to complete projects in VR.

In this preliminary work, we collect and analyze VR projects provided by the hosting and control version service "Github".

[1] data from http://steamspy.com/app/457320

We provide a mapping of existing VR projects in terms of Github properties, development environment, file distribution, etc. Since this area is not mature enough, and developers are still in the testing and exploration phase of these technologies, an evaluation of the design and code quality of the projects is also performed to draw a picture of the state of VR applications development. Additionally, we conduct a complementary study on Stack Overflow to identify the main themes that are of interest or represent challenges for VR developers.

Our study investigates four research questions corresponding to four layers of analysis:

(RQ1) What is the profile of VR projects on Github?

(RQ2) What are the most popular languages and game engines used in VR projects?

(RQ3) What is the quality of the projects from a perspective of software metrics?

(RQ4) As mined from StackOverflow, what are the main topics of interest for VR developers?

The remainder of this paper is organized as follows. Section II provides background information about Virtual Reality, describes the most popular virtual reality development platforms, and introduces the software metrics used in our study. Section III presents the design of our study, discussing the rationale behind our selection of VR projects and StackOverflow questions. Section IV presents and discusses the results of our exploratory study. Section VI discusses the limitations of our study. Section VII reports about related works. Finally, Section VIII concludes the paper and sets our perspectives.

II. Background

In this section, we give background information for our study. First, we briefly describe VR and some main VR devices. Then, we discuss popular VR development platforms and finally, introduce the software quality metrics used in this paper.

A. Virtual Reality

Virtual reality or computer-simulated reality refers to computer technologies that use software to produce realistic images, sounds and other sensations that replicate a real environment. Ideally, a VR experience will involve its user to the point that he feels physical presence through the ability to look and move around in the artificial world, and interact (sometimes with haptic feedback) with virtual elements displayed on a screen or in goggles. Although virtual realities can be displayed on computer monitors or projector screens, VR is much more compelling when experienced through virtual reality headsets (also called head-mounted displays or HMD) that succeed in creating an illusion of three-dimensional depth. In these settings, the user's vision is entirely confined within the virtual reality, making for very immersive simulations. Virtual Reality has thus been defined as "...a realistic and immersive simulation of a three-dimensional environment, created using interactive software and hardware, and experienced or controlled by movement of the body" [1] or as an "immersive, interactive experience generated by a computer" [2]. Applications of the technology seem limitless and range from primarily video games to fields such as medicine, entertainment, military, education, etc.

Recent years have seen a real surge in interest in VR, following the (pre- and actual) releases of groundbreaking (i.e., consumer-ready and affordable) HMDs like the Google Cardboard (and its plethora of clones), the Oculus headsets (SDK1, SDK2 and ultimately the Rift), the Vive from HTC, and the Samsung Gear VR from a partnership between Samsung and Oculus. In this regard, 2016 has been a pivotal year with the official commercial release of most of these HMDs. In particular, in this era of new-generation HMDs, March 28th, 2016 is an important date as it marks the official release of the first-to-market Oculus Rift in the defining category of consumer-grade computer-hooked HMDs. Next to this category of high-end VR devices, are even more popular (because cheaper and less computationally demanding) devices that run on smartphones.

B. Virtual reality development platforms

Different kinds of software are available to create VR applications. Developers can directly rely on the native software development kits (SDKs) or take advantage of the many capabilities offered by game engines and frameworks. Another interesting avenue for VR development is offered by new web browsers intended for VR [3].

Native software development kits (SDKs) which are device drivers and software libraries provide direct interactions with the operating system. Although it is possible to rely solely on these SDKs, most developers will use game engines to handle 3D graphics, physics and, if applicable, game behaviors. These frameworks often have strong cross-platform support, and usually come with very helpful tools like level editors or integrated development environments (IDEs). The best known in this category are the **Unity engine** and the **Unreal engine** (not only for VR development, but also for AR development, 2D, etc.). On the web side, Web browsers are arguably very much in sync with new technologies for VR development. Technologies like HTML5, WebGL, and JavaScript allow developers to create cross-platform VR applications. Finally, 360 degree, stereoscopic video captured from the real world (then possibly enhanced) can make for particularly realistic experiences, even though they are not fully interactive in the way that a 3D virtual environment can be.

In the next subsections, we summarily present key dominant game engines.

1) Unity Game Engine: Unity is a multi-platform game engine (smart-phone, Mac, PC, video game consoles and web) developed by Unity Technologies. It is offered under a free license and another professional paid one. It offers rapid prototyping and a possibility to deploy developed applications on different VR devices. It allows easy generation[2] of VR applications for most popular OS (including Windows, iOS, Android), most major gaming consoles, and the web. In

[2]For simple projects, it is as easy as clicking on a VR checkbox

addition, it supports several 3D formats and various audio and video resources. All this makes it one of the most popular engines on the market. There are more than 250,000 subscribers on the Youtube channel “Unity" who follow tutorials, explanations, compilations of the best games and demonstrations of performance and capabilities offered by the Unity game engine. Another channel “Unity asset store" focuses on the best assets[3] published and realized with Unity and has more than one million subscribers.

2) Unreal Game Engine: Unreal is developed by Epic Games (an American video game development studio). Currently it is free and in its fourth version. It is a multi-platform game engine that contain programming guides (best programming practices), demos and video tutorials, etc. The Unreal application market exposes various paid applications and free demos. To submit an application for sale in this market repository, the Epic Games team requires a certain level of quality and provides a detailed report of the process in a submission guide. In addition, it requires paying 5% of the gain for each application sale.

C. Software quality metrics

Mordal-Manet et al. [4] proposes to measure variables such as project size, complexity, cohesion, coupling and inheritance, to understand the quality characteristics of software projects. To measure these attributes, one needs to measure metrics associated with each attribute. There are two types of metrics to consider: primitive metrics related to the basic properties of the source code (such as the number of lines of code, the cyclomatic complexity, and so on) and design metrics that determine whether good practices and design principles were followed during the development of the project, e.g., whether a low coupling and a high cohesion is maintained.

To estimate a software quality attribute, for example the code reusability, which is defined as “the ability of software to integrate new implementations" [ISO 9126] [5], we should compute a set of metrics such as the number of Source Lines Of Code (SLOC), the Cyclomatic Complexity (CC), the number of methods per class and the Depth of the Inheritance Tree (DIT), which are necessary to estimate the code reusability [4]. Since VR applications are not all developed using object-oriented programming languages, we also use the number of functions to capture projects size.

Several tools have been created to automate the computation of metrics [6], [7], [8]. Among these tools, we selected *Understand* [9] which is a multi-platform tool for code analysis that support a large number of programming languages, including C, C++, C#, Java, Python, Objective C, PHP, and JavaScript. Using this tool, we are able to measure size, complexity, coupling, cohesion, and inheritance. Also, the level of documentation and comprehensibility of the code can be assessed by measuring the number of comment lines. In summary, the metrics and attributes that are considered in this study are the metrics proposed by Ammar et al. [10], with consideration to the limits of the used tool. Table I summarizes the relation between the quality attributes and the metrics collected using the software $Understand$ to evaluate the quality of VR projects. We provide the definition of each of our studied metrics below.

1) **Primary metrics :**
   a) Size,
      i) Number of Lines Of Code (LOC),
      ii) Number of Classes,
      iii) Number of Methods,
      iv) Number of Functions,
      v) Number of Code files,
   b) Complexity,
      i) Cyclomatic Complexity (CC),
2) **Design Metrics :**
   a) Inheritance,
      i) Depth of Inheritance Tree (DIT),
      ii) Number Of Children in Tree (NOC),
   b) Coupling,
      i) Number of coupled classes,
   c) Cohesion,
      i) Percentage of lack of cohesion (LCOM),
3) **Documentation and comprehensibility :**
   a) Number of comments lines,

III. Study Design

In this paper, we perform an exploratory study on open source VR projects from GitHub, which is the largest code repository hosting site in the world [2], in order to answer our research questions.

Additionally, we complement the study with an analysis of VR-related Stack Overflow questions, leading to

RQ4: As mined from StackOverflow, what are the main topics of interest for VR developers?

A. GitHub Analysis

Our main focus is on the exploration, from both project and source code perspectives, of virtual reality projects. To do this, we follow the systematic approach recommended by Basili *et al.* [11].

In the first step, we filtered and collected VR open source projects that matched the keywords “Virtual Reality" and “VR" using the search tool of GitHub. To separate the signal (repositories containing engineered software projects) from the noise (e.g., home work assignments) on GitHub repositories, we adopted a stargazers-based classification [12], and only considered projects with GitHub *stars* greater or equal to three ($stars :>= 3$). In short, we used the threshold $stars >= 3$ to filter out toy projects, following Munaiah’s *et al.*[12] observation that only a very small percentage ($\approx 11\%$) of scored GitHub repositories contain engineered software projects.

[3]An asset file is a file containing textures, shaders, and other related data for 3D models. These files are created when a new game project is compiled.

Table I
RELATIONSHIP BETWEEN QUALITY ATTRIBUTES AND METRICS

Property/Attribute	Size	Coupling	Cohesion	Inheritance	Complexity
Flexibility				X	
Maintainability		X	X	X	X
Comprehensibility		X	X	X	
Reusability	X		X	X	X

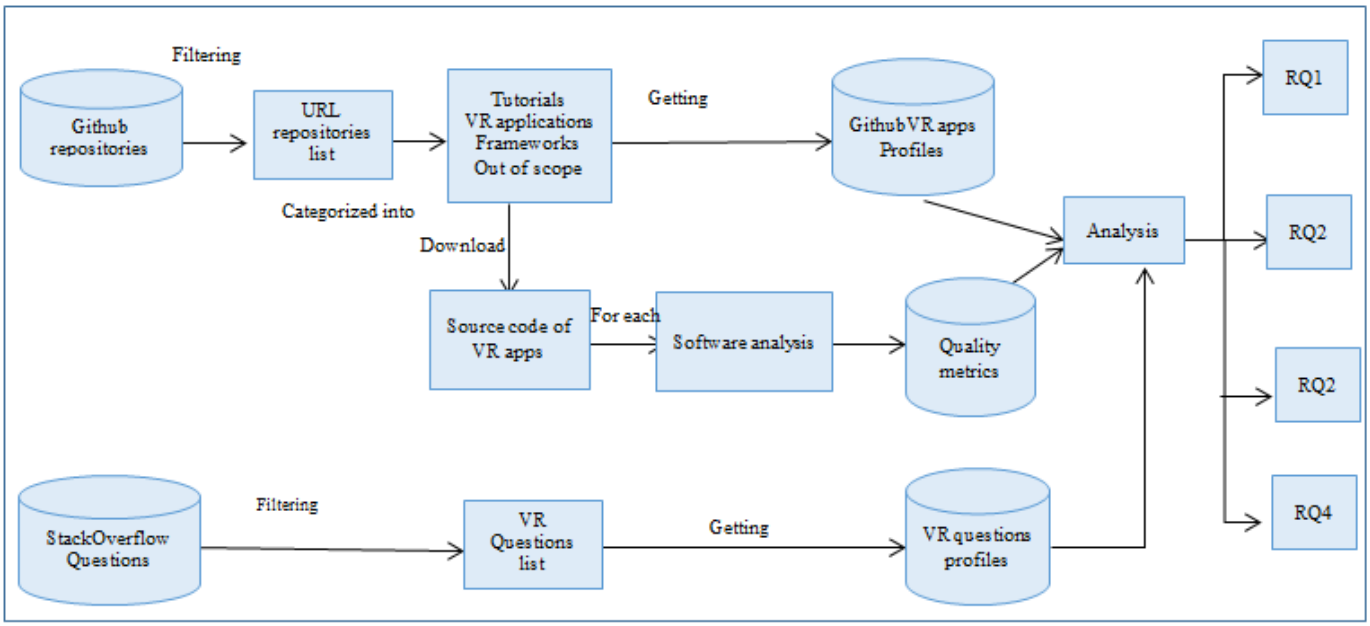

Figure 1. Overview of our research approach.

As a second criterion, we considered the status of the projects i.e., whether they are active or not after March 28th, 2016, which is the release date of the first commercial version of a computer-hooked HMD. Projects without new submissions after March 28th, 2016 were removed[4].

After these filters ($stars :>= 3 \,\&\, pushed :>= 2016-03-28$), we analyzed the projects' description as expressed in their "README.md" files to keep only actual VR applications. We eliminated redundancies in the lists of projects obtained using the two queries ("$Virtual\ Reality$" & $stars :>= 3 \,\&\, pushed :>= 2016-03-28$ and "VR" & $stars :>= 3 \,\&\, pushed :>= 2016-03-28$). Finally, we categorized the projects as VR applications, frameworks, tutorials or out of scope. As a result of this process, we obtained a final sample of **320 VR applications**.

The analysis of the obtained sample of projects was conducted as follows. First, we sought some insights about the GitHub profile of each project and collected for each project, its number of *watchers, stars, forks, issues, pull requests, commits, branches, releases, and contributors*. A second analysis was focused on determining which game engines were the most used. A third analysis was conducted at the file level: it consisted in collecting data on files' type (source code, images, audio, video, and others) and aimed at revealing the most used programming languages in VR applications. The final analysis on Github VR projects focused on evaluating the quality of VR projects through the metrics discussed in Section II and obtained using the *Scitools Understanding*[5] analysis tool.

B. Stack Overflow study

To understand issues faced by VR developers, we queried the Question and Answer (Q&A) website Stack Overflow (SO), seeking questions and answers exchanged by developers working on VR applications. We selected this Website because it is the dominant technical Q&A platform for software developers, with 49.2 million monthly unique visitors[6] (as of October 2017). We queried SO for questions that are tagged "VR" and "virtual-reality" using the search engine of SO. As additional filtering criteria, we only retained questions that have at least one answer and a score of at least one. This filtering is important to ensure that the retrieved questions are relevant for the VR community. The first query ("$virtual -$

[4]Note that we ended our collection of VR projects on September 30th, 2017

[5]Available on https://scitools.com

[6]https://www.quantcast.com/stackoverflow.com\#/

$reality \,\&\, answers \;:>=\; 1 \,\&\, score \;:>=\; 1''$) returned 193 questions while the second query ("$vr \,\&\, answers \;:>=\; 1 \,\&\, score \;:>=\; 1''$) gave us 1024 questions. Following the same filtering date criteria for GitHub, we took only in consideration questions asked on SO between March 28th, 2016 and September 30th, 2017. Finally, we manually analyzed the descriptions of all the obtained questions and eliminated the questions that were not related to virtual reality[7]. We collected the following SO properties for each question in our final data set: associated tags, score, number of views, number of answers, date of the question.

IV. Results

In this section, we report and discuss the results of our study; answering our four research questions.

A. *RQ1: What is the profile of VR projects on Github?*

We computed the metrics described in Section II-C on the 320 VR projects downloaded from GitHub. For each project, we also extracted: (1) the project's proprieties as exposed on Github (number of stars, commits, forks, pull requests, issues, releases, contributors, watchers, and branches); and the (2) distribution of files by type (as discussed in Section III-A).

Table II
Number of VR projects per property of GitHub. Reading example: 57 analysed projects had exactly 1 fork (line "Forks", column "1").

Property \ Interval	0	1	]1..10]	[11..100]	>100
Forks	80	57	139	43	1
Pull request	291	21	7	1	0
Issues	212	46	43	19	0
Commits	0	4	104	159	53
Releases	260	18	31	11	0
Branches	0	201	110	9	0
Stars	0	0	184	124	12
watchers	5	57	197	59	2
Contributors	0	189	128	3	0

Table II summarizes the results about GitHub properties. Overall, we can conclude that in VR projects:

1) Projects are mostly developed individually: 59% of projects have only one *contributor*. Interestingly, compared with the numbers (67-72%) reported in [2], this would suggest that VR projects on Github are actually slightly more collaborative than the average Github project.
2) *Pull requests*, which are arguably the preferred way to submit contributions to a Github project, are quite rare: only 9% of the projects. This is still three times more than the roughly 3% reported overall for Github projects in [2].
3) Most projects are immature : 81% of VR projects have no release.
4) Projects are developed mostly in one *branch*: 63% of projects have a single branch.
5) The most common *commits* range is between 11 and 100 commits (signaling a low-to-moderate development intensity): about 50% of projects hold a number of commits in that interval.

[7]A majority of questions with the vr tag were unsurprisingly not about virtual reality

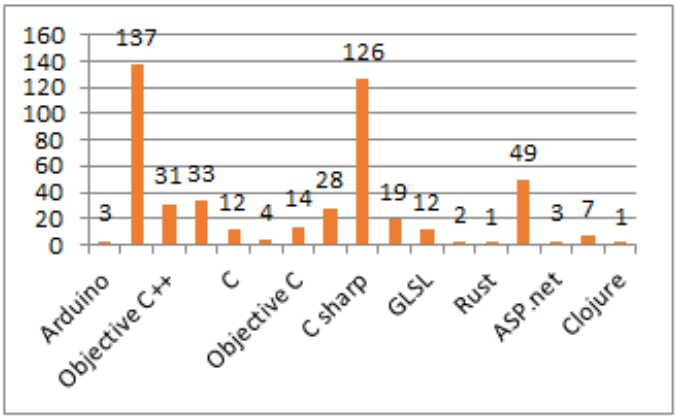

Figure 2. Programming languages used in VR projects (JavaScript is the top language: 137 projects)

A second layer of our analysis of VR projects on GitHub focuses on the kind of files (source code, asset, image, audio, video or others) contained in the projects. We found that (1) VR projects contain 15% of source code files; (2) 7% of files are images; (3) 2% of files are assets; (4) 2% of files are audio or video, and (5) 74% are others such as JSon files, XML configuration files, HTML files, shader and texture files, etc.

> *Overall, our results show that most VR projects hosted on GitHub are developed by lone developers working in a single branch of development. A number of other observations, such as the rarity of Pull Requests and issue filing, may stem from that fact. The projects have a small number of releases, commits, watchers and forks. We found only three projects with more than 10 contributors.*

B. *RQ2: What are the most popular languages and game engines used in VR projects?*

We analysed the number of projects by programming language. Our results, presented in Figure 2 show that JavaScript and C# are the most popular languages for VR development (82% of projects). Javascript is used in 43% (137/320) of projects, followed by C# in 39% (126/320) of projects. It should be noted that both languages can be used in Unity. In particular, C# is the main language of Unity users.

Finally, we analysed the projects to identify the number of projects that use the two popular game engines (Unity and

Unreal). We found that 33% of projects use Unity and only 3% use Unreal.

Our results show that the most popular languages are JavaScript and C#, and Unity is the most used game engine during VR application development.

C. *RQ3: What is the quality of the projects from a perspective of software metrics?*

To answer this research question, we resorted to analyzing software metrics from seven perspectives (size, complexity, inheritance, coupling, cohesion, comprehension) in order to get some insights about the quality of VR projects – at least from a source code point of view. Using the *Understand* software static analysis tool, we were able to analyze 297 projects out of 320; 23 of the 320 projects could not be analyzed because they were developed using programing languages that are not supported by the tool. We summarize our key findings below.

1) Project size: To evaluate VR projects' size on GitHub, we computed the Source Lines Of Code (SLOC) metric. In order to classify the project sizes, we used the classification proposed by Boehm [13] which categorizes a size of KLOCs (Thousand Lines Of Code) as small, intermediate, medium, wide and very wide. Based on this categorization, our intervals will be expressed in KLOC as follows :

- Small from 0 to 2
- Intermediate from 2 to 8
- Medium of 8 to 32
- Wide from 32 to 128
- Very wide from 128 to 512
- Extremely wide if it exceeds 512

Results on Table III show that 27% (79/297) of projects are small projects that have less than 2 KLOC. 15% (44/297) of projects have an intermediate size of SLOC. A significant number of projects are medium to wide sized projects, i.e., 20% of projects are of a medium size, 23% are of wide size, and 13% of projects are of very wide size. Only six projects are considered as extremely wide.

Table III
OVERVIEW OF TSLOC INTERVAL AND VR PROJECT SIZES

Categorisation	Interval of TSLOC	Number of projects
Small	[0..2]	79
Intermediate	]2..8]	44
Medium	]8..32]	60
Wide	]32..128]	70
Very wide	]128..512]	38
Extremely wide	]128..512]	6

Another approach to measure the size of VR projects is to count the number of classes, methods and functions per project. Table IV summarises the number of classes, methods and functions in the studied projects.

Table IV
NUMBER OF PROJECTS PER INTERVAL OF FUNCTIONS, METHODS AND CLASSES

Interval/Number	Number of projects by interval of classes	Number of projects by interval of method	Number of projects by interval of function
[0...10]	111	4	111
[11...100]	94	85	45
[101..500]	62	47	41
[501...1000]	15	22	36
[1001...10000]	15	123	61
>10000	0	16	3

Analyzing our results, we observe that most of the projects are small: (1) 37% (111/297) of projects have ten or less classes; (2) 35% have 100 or less methods; (3) only 5% of the projects present more than 10 000 functions and they generally followed the Functional development paradigm in JavaScript.

Our results on the size of VR projects corroborate our previous observation that the majority of VR projects on GitHub are small to medium projects. Furthermore, bigger projects do not use the Object Oriented paradigm. They adopt Procedural or Functional paradigms in JavaScript.

2) Project complexity: We computed the McCabe's Cyclomatic Complexity (CC) [14] in order to assess the complexity of VR projects. Although not an absolute requisite, a CC below 10 is generally recommended. We analyse the extracted complexity metrics through that lens. Table V shows the distribution of projects with CC>10. We found that 30% (91/297) of projects do not have a method with CC>10. In total, projects that have at least 11 methods with CC>10 account for 47% (139/297). These results would suggest that a slight majority of the VR projects exhibit acceptable complexity.

Table V
NUMBER OF PROJECTS PER METHOD INTERVAL WITH CC>10

Interval / Number of methods or function with CC>10	Number of projects
0	91
[1..10]	67
] 10..100]	100
] 100..1000]	39

3) Projects inheritance: In this study, inheritance is measured by two metrics: the Number Of Children in Tree (NOC) and the Depth of Inheritance Tree (DIT).

For NOC, we found that 43% of projects exceeded the limit of 10 children [15] in a class (misuse of sub-classifications),

with numbers of children ranging from 11 to 1705 (Figure 3)[8]. For the DIT, which indicates the balance between reusability and complexity of the class, we found that most of VR projects have a DIT less or equal to 4 (respect of encapsulation). Only 5 projects exceed this number with 8 as a maximum value.

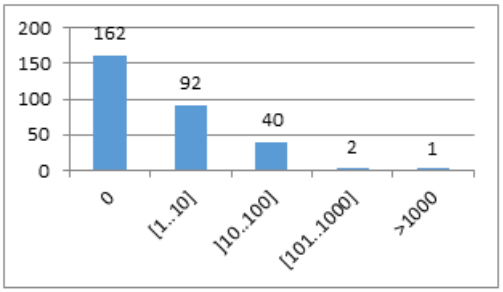

Figure 3. Number of VR Projects per Number Of Children in Tree (NOC) Interval

Table VI
NUMBER OF PROJECTS PER MAXIMUM COUPLED CLASSES(MCC) IN A PROJECT INTERVAL

Interval / Number of maximum coupled classes	Number of projects
0	118
[1..10]	49
] 10..100]	130
>100	0

4) Projects coupling: Coupling metrics capture the degree of interdependence between the classes/modules of a project. It is desirable to maintain a low coupling in a project since a high level of coupling is reported to have a negative impact on software maintenance [16]. Results returned by Understand about the Maximal Coupled Classes (MCC) per project (Table VI) indicate that approximately 40% (118/297) of VR projects do not have coupled classes. Sixteen% of the VR projects exhibit a reasonable maximum coupling, ranging from 1 to 10 but another 44% presents potentially problematic levels of coupling (up to a MCC of 89).

5) Projects Cohesion: Cohesion refers to the degree to which the elements of a module belong together [17]. The level of cohesion of a class is considered problematic if the class has more than 10 methods with a LCOM greater than 80% and possess at least 10 variables. In our dataset we found 95 projects (31% of projects) having at least one problematic class. The worst project, with respect to this metric, has 11 classes with problematic levels of cohesion.

6) Projects comprehension: The metric chosen to measure the level of code documentation is the number of comment lines. Table VII shows that 74% (219/297) of VR projects have less than 20% comments, which is a commonly recommended threshold.

[8]In the figure, we can see that there are 40 projects with NOC between 10 and 100, 2 projects with NOC between 101 and 1000, and one project with NOC>1000 (exact number being 1705).

Table VII
NUMBER OF VR PROJECTS PER PERCENTAGE OF COMMENTS LINES

Percentage of comments	Number of projects
[0...20%]	219
[20%...40%]	60
>40%	18

Most of the current GitHub VR projects are small to medium projects. They can be somewhat complex and do not always have enough comments to ensure good comprehensibility. However, most of the projects make a proper use of inheritance and have a good level of cohesion. More than half of the projects have a reasonable level of coupling. Hence, we conclude that current VR projects on GitHub are of decent quality.

D. *RQ4: As mined from StackOverflow, what are the main topics of interest for VR developers?*

We identified a total of 297 VR-related questions on Stack Overflow. For each of these questions, we collected data pertaining to the date on which the question was asked, the question score, its number of views and answers, and its associated tags. In this research question, we are interested in understanding issues faced by VR developers. Hence, we first grouped the questions according to their tag. Tags are used in SO to capture the topics of questions and answers.

1) Dominant Tags: We computed the number of questions in each tag category and found the number one tag to be unity (122 mentions), followed by android (83 mentions) and javascript (76 mentions). These are followed by Google-related tags such as google-cardboard (68) and google-vr (64). There is a significant co-occurrence between the google-related tags google-cardboard and google-vr, making issues about google-related technologies the dominating theme about VR developers, i.e., 156 questions in total. VR developers also asked multiple questions about JavaScript frameworks like aframe (36) and webvr (34) and the language c# (33). Additional details on dominant tags for VR questions on SO can be found in Table VIII

Another way to look at the collected data is to regroup tags according to their types. We considered the following categories: frameworks, languages, operating systems and platforms, HMDs, technology companies. Tags that did not fit in any category, typically terms related to aspects that are less focused on software engineering, were labeled as “Others”. Table IX summarizes our findings. We can observe that unity, javascript, android, google-cardboard and google largely dominate their respective categories. On frameworks, it should be noted that javascript-based frameworks are collectively almost as important as unity; unreal is barely mentioned, with only 3 questions with that tag. Company-wise, Google is a dominant

Table VIII
DOMINANT TAGS IN VR-RELATED QUESTIONS ON SO – SORTED BY THE NUMBER OF MENTIONS

Tag	Occurrences	Score	Views
unity	122	208	61,682
android	83	178	34,015
JavaScript	76	135	28,428
cardboard	68	117	36,878
google-vr	64	143	38,774
aframe	36	53	10,010
webvr	34	69	18,041
c#	33	54	19,227
oculus	27	45	8,105
ios	23	36	6,898
react	21	52	3,801
gear-vr	15	25	4,884
three.js	14	20	4,165
daydream	13	22	3,974
video	12	18	5,348
vive	10	18	3,613

player, with Android and Cardboard being clearly relevant to VR developers. Apple technologies are taking a back seat with ios and objective-c clearly behind in their respective categories.

2) Trend Analysis: To further understand the gathered data, we grouped it per quarter, starting from April 2016[9] and ending on September 30th, 2017. We then study the results obtained on the 6 quarters: Q1 grouping the first three months (April to June 2016) and Q6 the last three months (August to September 2017).

First, we looked at the numbers, just on the basis of the number of questions asked, their cumulative score and views. Table X presents data about VR-related questions on SO through quarters 1 to 6. A first observation is that the last quarter (Q6) presents significantly lower numbers. Whether this previews declining interest for VR or signals maturing VR technologies (with more documentation and tutorials) is up to debate. In any case, we considered that last quarter as an outlier and focused our analysis on quarters 1 to 5. As a general observation for those quarters, except for views (which were at their peak just after the launch of the new generation PC-hooked HMDs), there is a relative stability in the numbers.

Next, we considered individual tags through Quarters 1 to 5 in a bid to determine which tags are trending up and which ones are trending down. Table XI presents data about these trends for tags with enough mentions throughout the quarters. In the framework/engine category, questions on unity and google-vr appear relatively stable. Trending up are webvr (from 3 at Q1 to 12 at Q5) and react, which relevance for VR shot up to 15 mentions in Q5 (from only 3 questions throughout Q1 to Q4), with the release of the Facebook and Oculus-backed framework React-VR. Also trending up is three.js (from an average of 2 questions in quarters 1-4 to 7 in Q5).

Language wise, JavaScript definitely trends up from 5 in Q1 to 35 in Q5 while C# appears to be trending down. As for the other languages, there is not enough data to speculate either way. On operating systems, Android appears slightly down while iOS is stable. Device-wise, the cardboard appears somewhat down from its peak in Q1 (22) to only 12 in Q5, possibly due to the emergence of the Daydream, the newer Google VR device. Other devices like the Gear VR and the HTC Vive appear stable. Finally, company-wise, Google is consistently the biggest player, Oculus appears slightly down and Apple appears stable.

[9]including results from March 28, 2016

V. DISCUSSIONS

To the best of our knowledge, our report is one of the first to analyse VR projects in a Software Engineering perspective. Our results show that most open source VR projects today are developed by lone individuals. Although code sharing and issues' resolutions are usually done nowadays through pull requests (in order to have an agreement before integrating new pieces of code), this practice is mostly absent in VR projects, We found that only 29 projects contain from 1 to 14 pull requests. A large number of VR projects have a number of forks that vary between 1 and 10, which could be a hint that very few code reuse activities occur during VR project developments. JavaScript and, to a lesser extent, C# are the dominant languages for the development of VR applications.

VR open source developers have a strong preference for the game engine Unity over Unreal. This dominance is explained by different reasons. First, Unity offers the possibility of coding, among others, in C# and JavaScript languages. Moreover, previous analysis of the programming languages used in the selected VR applications show that programmers have more interest in coding in JavaScript followed by the C# language, while Unreal Engine 4 uses only the C++ language. Additionally, Unity's asset store appears to be broader and richer than Unreal store. It provides developers with functionalities from simple and intuitive animation, up to providing GUI generators and motion capture software .

The metrics of Understand analysis tool confirm that the majority of projects have small to medium size according to Lines of codes (only 38% of projects are considered as wide to extremely wide). Cyclomatic complexity metrics show that close to half of the VR projects have at least one complex method. However, the levels of coupling and cohesion are reasonable. Of concern is the relatively weak level of commenting, with 74% of projects presenting at most 20% of comments in their souurce code.

The complementary study on Stack Overflow reinforces the above observations. On game engines, Unity is indeed a major topic in VR while Unreal is much less asked about. As for languages, Javascript is the dominant and growing topic

Table IX
DOMINANT TAGS IN SO, BY CATEGORY AND SORTED BY DESCENDING ORDER OF MENTIONS

Frameworks	Languages	OS/Platform	Devices	Company	Others
unity (122)	javascript (76)	android (83)	cardboard (68)	google (156)	video (12)
google-vr (64)	c# (33)	ios (23)	gear-vr (15)	oculus (27)	panorama (8)
aframe (36)	java (7)	chrome (5)	daydream (13)	apple (25)	camera (6)
webvr (34)	objective-c (4)	windows (3)[*]	vive (10)	htc (10)	–
react (21)	swift (3)	–	–	–	–
three.js (14)	–	–	–	–	–

Table X
STACK OVERFLOW - TRENDS FOR VR RELATED QUESTIONS

	Q1	Q2	Q3	Q4	Q5	Q6	Total
Mentions	55	54	49	52	56	29	295
Score	101	106	87	106	103	40	543
Views	50K	24K	22K	21K	14K	2K	132K

Table XI
STACK OVERFLOW - VR RELATED TAGS TRENDS (MENTIONS)

	Q1	Q2	Q3	Q4	Q5
google	37	30	24	27	31
unity	23	18	23	27	23
android	21	17	13	13	15
javascript	5	12	10	11	35
cardboard	22	14	11	7	12
google-vr	12	13	12	17	10
aframe	1	10	6	5	14
webvr	3	5	7	6	12
c#	5	3	5	11	6
oculus	8	6	3	4	5
apple	6	5	3	5	4
javascript	2	3	4	2	11
ios	5	4	3	5	4
react	1	1	0	1	15
gear-vr	3	3	2	4	2
three.js	0	2	2	3	7
daydream	0	0	6	3	4
vive	0	0	2	4	4

followed by C#, which is a distant second. Other languages are barely mentioned. The SO study provides additional insights as to the dominant target OS (android) and device (cardboard), both pointing to a clear dominance of Google technologies.

VI. LIMITATIONS

The main limitation of this work relates to the fact that data used in the reported exploratory study was gathered from only GitHub. Open source projects on GitHub represent a partial sample of the VR project population. In fact, commercial and closed source VR projects may follow different development practices than open source VR projects. Unfortunately they are intrinsically inaccessible for researching at the moment. Furthermore, selected criteria reduced the sample to only a fraction of the existing projects on Github. New projects that were not stargazed yet were ignored. These limitations are somewhat mitigated by the addition of a complementary study on Stack Overflow. However, that study also makes use of filtering criteria that could skew its results as questions with a score of 0 or no answers may contain more than pure noise.

There is an internal validity threat related to our use of the *Understand* static analysis tool to calculate metrics and assess code quality. This tool does not support all programming languages and some relevant quality metrics are not provided. More analysis should be conducted with other tools.

Nevertheless, we believe that the results reported in this paper will contribute to raise the awareness of the SE community about the needs of VR software developers. Further analysis with other open and close source projects are necessary to deepen our understanding of the state of practice of VR applications development. We provide with this report a repository [10] containing the raw data used for this study.

VII. RELATED WORK

In this section, we discuss related work that investigated Virtual Reality projects as well as works conducted on Stack Overflow and GitHub.

A. Virtual Reality

Despite the fact that Virtual Reality has been a research topic for decades, to the best of our knowledge, there is only one other preliminary study that investigated the software engineering aspects on VR software development. Rodriguez and Wang [18] proposed an empirical study of VR projects from Unity List, a repository of projects built using Unity 3D. They found that there is a growing number of VR projects, with a pace picking up significantly first on Quarter 4 of 2015, and then from Quarter 2 of 2016 (which is the starting period of our study). pic the number of open source VR software projects are steadily growing. Other findings are that some large projects are starting to emerge while games and

[10]https://github.com/empiricalVR/GITSO1

development on Oculus seemed dominant. Differently from [18], our study is not limited to Unity projects and also integrates a study of StackOverflow questions.

Similarly to [18], we propose a study aiming to highlight software engineering issues relevant, not neccesarily exclusively, for VR projects. In this perspective, three other studies are worth presenting. Pausch *et al.*[19] presented a detailed discussion about VR project development ("Alladin"), discussing challenges and providing suggestions to software engineers, producers, and authors. Capilla and Martínez [20] examined the use of architectural patterns to improve the development of complex VR systems. They also proposed new patterns proper to the VR domain. Finally, Elliott *et al.*[21] described a vision of affordances offered by VR, exploring some benefits of VR to support software engineering activities and discussing future work, open questions, and the challenges of VR.

B. Stack Overflow and GitHub

Q&A websites like Stack Overflow provide users with a platform where they can exchange knowledge and expertise. Since the introduction of Stack Overflow in 2008, a plethora of studies have examined knowledge sharing activities on this Q&A website, designed for developers. Anderson et al. [22] proposed models to predict the long-term value of Stack Overflow questions and answers. They also proposed models to determine whether a Stack Overflow question requires a better answer. Barua et al. [23] explored topics and trends on Stack Overflow and observed a significant growth of the share of questions related to mobile application development. They also observed a decline of questions regarding the .NET framework. Vasilescu et al. [24] investigated the interplay between Stack Overflow and the repository hosting website, GitHub. They observed that active Github committers ask fewer questions on Stack Overflow than other users. They also observed that questions on Stack Overflow are often associated to GitHub repositories.

Gousios et al. [25] explored the mechanisms of pull-based software development in GitHub and report that the pull request model offers a fast turnaround, quick code integration times, and more opportunities for community engagement (for example to the code review process). However, the results of our analysis in this paper show that VR developers on GitHub do not take advantage of these opportunities. They do not follow the pull request model.

VIII. Conclusion

In this work, we analyzed 320 Virtual Reality open source projects in a Software Engineering perspective. Our main contribution is unveiling the state of practice of open source VR application development, using Github's project profile attributes and software quality metrics. Additionally, we tried to reveal topics of interest of VR developers by analyzing Stack Overflow VR questions.

Our results show that most VR projects are developed by lone developers working in a single branch of development, and almost without pull request. VR projects on GitHub have a small number of releases, commits, watchers and forks. It is remarkable that only three projects have more than 10 contributors. Moreover, we observe that the most popular languages are JavaScript and C#, and Unity is the most used game engine during VR applications developments. Nevertheless, almost all the studied VR projects comply with recommended levels of inheritance and cohesion. More than half of the projects have a reasonable level of coupling between classes.

Our work shows that despite its potential to improve critical domains such as medicine, engineering, education and entertainment, the Virtual Reality Software Engineering is not yet matured. Modern engines (such as Unity 3D or Unreal) provide some directions and guidelines, but future works have to be performed to understand the true needs and requirements of VR software engineers. In the future, we plan to continue analyzing VR projects as well as Augmented Reality (AR) projects to deepen our understanding of developers' needs and enable the development of tools and techniques to assist them. In particular, there is a need for specific tools that can analyse the quality of VR applications, to verify if best practices of VR development are established and enforced.

References

[1] "Dictionary.com unabridged," Oct 2017. [Online]. Available: http://www.dictionary.com/browse/virtual-reality

[2] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 92–101.

[3] T. Parisi, *Learning Virtual Reality: Developing Immersive Experiences and Applications For Desktop, Web and Mobile*, 2nd ed. Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc., 2016.

[4] K. Mordal-Manet, J. Laval, and S. Ducasse, "Modèles de mesure de la qualité des logiciels," in *Évolution et rénovation des systèmes logiciels*, Hermès, Ed. Hermès, Nov. 2011. [Online]. Available: https://hal.inria.fr/hal-00639279

[5] "Software engineering - product quality, ISO/IEC 9126-1," International Organization for Standardization, Tech. Rep., 2001.

[6] B. Baldassari, "Squore: a new approach to software project assessment." in *International Conference on Software & Systems Engineering and their Applications*, 2013.

[7] P. Cousot, "The role of abstract interpretation in formal methods," in *Fifth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2007)*, Sept 2007, pp. 135–140.

[8] N. Rutar, C. B. Almazan, and J. S. Foster, "A comparison of bug finding tools for java," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*. IEEE, 2004, pp. 245–256.

[9] SciTools, "Software understand," https://scitools.com/, 2017.

[10] S. R. Ragab and H. H. Ammar, "Object oriented design metrics and tools a survey," in *Informatics and Systems (INFOS), 2010 The 7th International Conference on*. IEEE, 2010, pp. 1–7.

[11] V. R. Basili, R. W. Selby, and D. H. Hutchens, "Experimentation in software engineering," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 7, pp. 733–743, July 1986.

[12] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating GitHub for engineered software projects," *Empirical Software Engineering*, pp. 1–35, 2017.

[13] B. W. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, pp. 4–21, Jan 1984.

[14] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, Dec 1976.

[15] C. Metzner, L. Cortez, and D. Chacín, "Using a blackboard architecture in a web application." *Issues in Informing Science & Information Technology*, vol. 2, 2005.

[16] A. B. Binkley and S. R. Schach, "Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures," in *Proceedings of the 20th International Conference on Software Engineering*, ser. ICSE '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 452–455. [Online]. Available: http://dl.acm.org/citation.cfm?id=302163.302212

[17] E. Yourdon and L. L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1979.

[18] I. Rodriguez and X. Wang, "An empirical study of open source virtual reality software projects," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, 2017, pp. 474–475. [Online]. Available: https://doi.org/10.1109/ESEM.2017.65

[19] R. Pausch, J. Snoddy, R. Taylor, S. Watson, and E. Haseltine, "Disney's aladdin: First steps toward storytelling in virtual reality," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 193–203.

[20] R. Capilla and M. Martínez, *Software Architectures for Designing Virtual Reality Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 135–147. [Online]. Available: https://doi.org/10.1007/978-3-540-24769-2_10

[21] A. Elliott, B. Peiris, and C. Parnin, "Virtual reality in software engineering: Affordances, applications, and challenges," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, May 2015, pp. 547–550.

[22] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: a case study of stack overflow," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 850–858.

[23] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[24] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Proceedings of the International Conference on Social Computing (SocialCom)*. IEEE, 2013, pp. 188–195.

[25] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 345–355. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568260