Programming Language Diversity and Software Growth: A Case Study

Samuel Barrett, Oscar Sandford, Natasha Bansal, Kenil Shah
University of Victoria, Victoria, Canada
{samarbarrett,oscarsandford,bansaln,kenilshah}@uvic.ca

ABSTRACT
Tu and Godfrey's seminal study on the growth patterns of Linux inspired scholars to look into the development of open-source software. Based on their methods as well as those of other studies, we analysed React Native, a project which uses multiple programming languages, to see how more language-diverse subsystems grow relative to subsystems with fewer languages. We found that the number of languages in a subsystem influences its growth rate by about 10%. We also note that some subsystems directly affect the growth rate of other, dependent subsystems. Our findings show that subsystems with more languages drive the growth of React Native. More research is needed to make conclusive points about the efficacy of programming language diversity across all open-source projects.

CCS CONCEPTS
Software and its engineering → Software evolution

KEYWORDS
software evolution, React Native, programming languages, case study

1 INTRODUCTION

Software is constantly evolving. Unlike other engineering constructs, software needs tuning, bug and vulnerability fixing, integration with other software, and a continuous stream of novel features fueled by market demand. It is inevitable that software under the pressures of pleasing all these avenues will grow as it evolves, even becoming bloated.

In 2000, Michael Godfrey and Qiang Tu published a seminal paper on the evolution of open-source software (OSS) projects. They analysed the Linux kernel project from both a system level and its major subsystems, which varied in size and complexity. Results showed that Linux was growing at a surprisingly super-linear rate over several years [1]. Our study leverages the methods of Tu and Godfrey in order to answer specific questions about software evolution in the modern day.

We investigated the effects of using multiple programming languages in a large, widely used OSS project. Using React Native as a case study, we tried to understand the motivations for using more than one programming language. Primarily, our goal was to understand how programming language diversity affects the rate of growth for individual subsystems with more language diversity.

We hypothesised that React Native components that make use of multiple programming languages would see less attention over time than components that primarily use a single programming language. As a result of this, React Native's growth will be stunted by the complexity inherent in a language-diverse environment.

2 BACKGROUND

React Native is a prime choice for a modern OSS project that is close enough to Linux in the

sense that it is widely-used and in development for over 5 years at the time of analysis. React Native differs in that it is hosted on GitHub (unlike Linux kernel code in the 1990's) and spearheaded by Facebook (now Meta). It is built to bring React's web UI framework to iOS and Android, using JavaScript, Java, Objective-C, C++, and more [2].

Our paper contributes insight on the efficacy of programming language diversity for OSS projects by using a case study project headed by a large company. We chose to bring Tu and Godfrey's quantitative analysis methods to light, as their results-based simplicity avoids the confounding variables of a hybrid approach. Such hybrid approaches involving a deeper look into the development practices and social environment of OSS projects present dangers of muddy observations through dirty data gathering, and biased interpretations of subjective percepts.

The methods we used yielded quantitative data, from which we drew our own conclusions. The reader is encouraged to complement our insights with their own interpretations of the data presented herein.

2.1 Related Work

A study by Karus and Gall analyzed the amount of code written in each language of a batch of OSS projects. They found that the amount of code written varies between programming languages [3]. Their results also highlighted the dependency of one programming language on another, an example being that JavaScript and CSS files evolve alongside XSL [3]. This understanding helped us in our analysis of subsystem dependency.

Tom Arbuckle's case study on `git` provided insight on the correlation of growth between programming languages. He used a normalised compression distance over each release patch in order to compute the information delta between patches [4]. Our metric considers growth rate across all releases instead of change between versions.

3 REACT NATIVE

In staying faithful to the methods of Tu and Godfrey, we examined React Native at a system level and at its subsystems. Subsystems are defined as directories that are direct children of the repository's root directory.

Unlike the original study, our data source is GitHub. Each stable release of React Native is marked with a specific branch, starting with 0.5-stable in June 2015 [2]. We cloned the repository and wrote Python scripts to `git checkout` each release's respective branch, up to 0.65-stable in November 2021.

Tu and Godfrey used the Linux utility `wc -l` to count lines of code. However, we decided to use `cloc` [5], as it is designed for differentiating modern language features. In a preliminary analysis, we noticed that `loc`, a similar tool, did not detect Ruby files. For this precision, we went with `cloc`.

As software evolves, subsystems are added and removed. Some are present through all releases, while others are only present in a subset of the releases. We counted lines of code (LOC) for each file in the subsystem, totalling the lines of code for each programming language used. We gathered this data for 37 total subsystems across 59 versions, as 0.16-stable is missing from GitHub [2].

3.1 Language Influence on Growth Rate

In order to capture the strength of language diversity in each subsystem over time, we simply averaged the number of languages present in each subsystem across every release. Our metric is defined as follows:

$$\left[\sum_{i=5}^{64}(LOC_{v(i+1)} - LOC_{v(i)})/(LOC_{v(i+1)} + LOC_{v(i)})\right]/$$

The scatterplot below shows correspondence between a subsystem's average language count and its growth rate in lines of code. A line has been fitted that shows a positive correlation between languages and growth, with an R^2 value of 0.1115, indicating that the number of languages explains about 10% of the subsystem growth rate assuming that the correlation is linear.

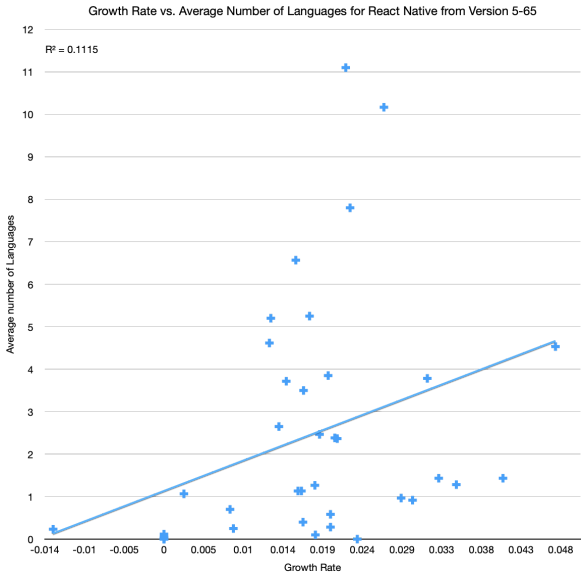

Figure 1: Subsystems with more languages tend to experience higher growth rates

3.2 Subsystem Dependencies

Initial analysis raised some concerns regarding the linked spike in growth of some of React Native's key subsystems around release 59, notably Libraries and React Common. The figures below highlight their LOC trends. It is worth noting that the "other" language category includes languages for files that were added and later removed in subsequent versions, such as Python or Assembly.

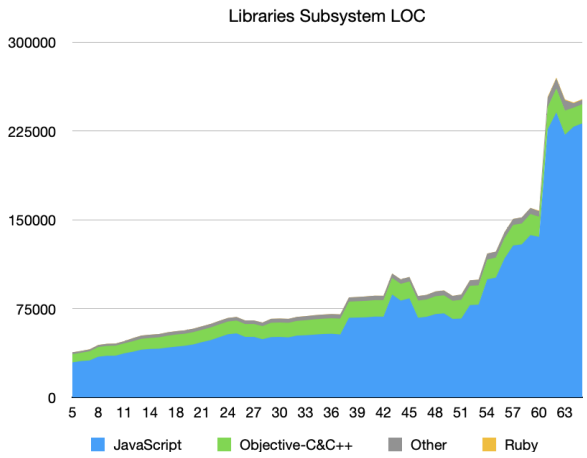

Figure 2: Total lines of code in Libraries, a subsystem with only two languages of consideration

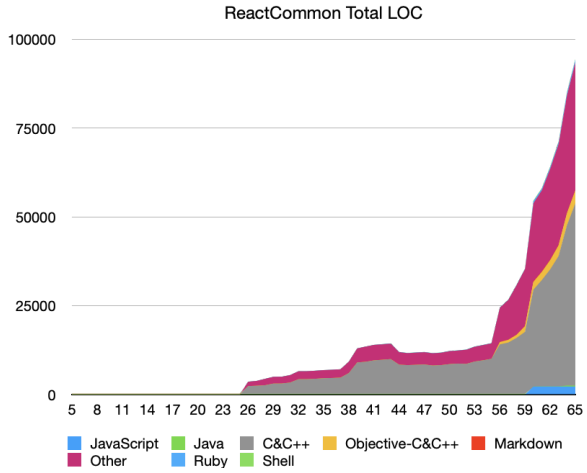

Figure 3: Total lines of code in React Common, a core subsystem with a multitude of languages

Both subsystems follow a similar trend in growth, so it is reasonable to say that they are correlated. The Libraries subsystem features predominantly JavaScript, with some Objective-C/C++. React Common is mainly C/C++ with a host of other supporting file types.

Other subsystems display flatter, consistent growth, such as React Android, hinting at their independence from subsystems like React Common. The Scripts subsystem's growth closely follows the overall growth of the system. Figure 4 and 5 below visualize this.

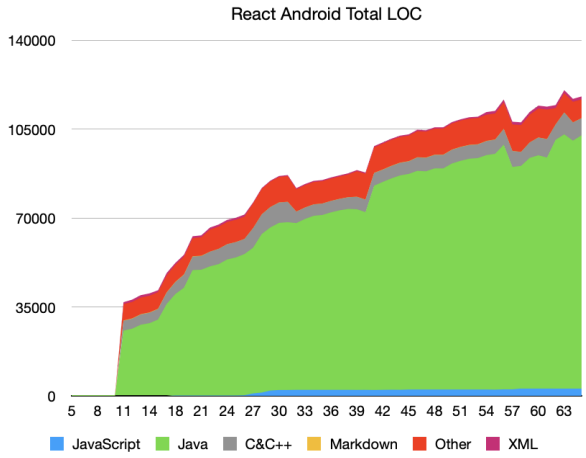

Figure 4: Total lines of code in React Android, a relatively independent subsystem with a few languages

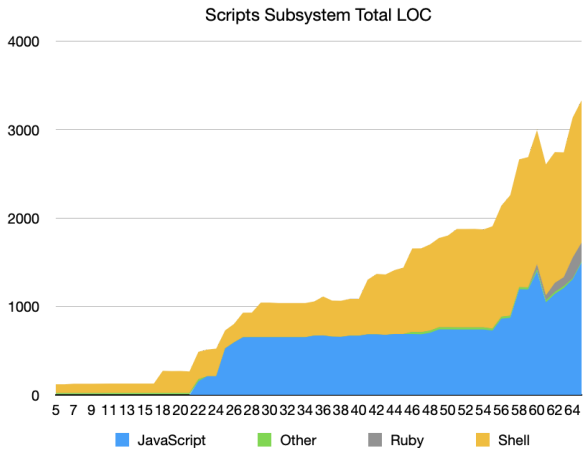

Figure 5: Total lines of code in Scripts, trending towards a mean between Libraries and React Android

4 DISCUSSION

In using Tu and Godfrey's methods and augmenting their analysis with our own metrics, we observed surprising results that suggest interesting correlations between subsystem growth, programming languages, and dependency.

4.1 Results Summarized

More programming languages inherently adds complexity, but the results from section 3.1 turned our hypothesis around. We supposed that, given Tu and Godfrey's results on subsystem complexity, complexity added through language would be a factor in slowing the development of these subsystems. However, our growth rate metric plotted with average language counts in Figure 1 shows that generally subsystems with more languages grow faster instead of slower.

In 3.2, we saw that this is most likely due to the fact that subsystems with more languages are developed more often, as they may be core components. As a core subsystem, React Common exhibits a diversity of programming language use, as well as a tendency to influence the growth of subsystems such as Libraries. Another reason could be that subsystems with more languages may be more decoupled, allowing different developers to work on different parts without too much effort.

Concerning cross-system influence, we observed that other technical subsystems like React Android are more independent from the growth effects in a core subsystem. Further, subsystems like Scripts that provide test suite support to the whole software find their growth following a smoother trajectory relative to the system as a whole.

4.2 Limitations

Our study is based on methods devised over 20 years ago in order to understand OSS growth. The original study used the command `wc -l` to count lines of code, whereas we used `cloc`. Minor inconsistencies may arise if this study is recreated using the `wc -l` utility.

Our methods only considered the content of React Native releases on GitHub at face value. In their ninth peril of mining GitHub for software projects, Kalliamvakou *et al.* warn that active projects may not conduct all their software development on GitHub [5]. Since React Native is an OSS project overseen by a big tech company, there is likely some internal development that was not captured by our

analysis of releases. In addition, React Native's external software dependencies were not considered. However, by considering releases instead of commits, we sacrificed granularity in order to avoid other perils related to the analysis of GitHub activity outlined in [5].

4.3 Threats to Validity

As this is a case study, confounding variables can present arguments against our results. This includes the potential for external development mentioned previously. Studies over multiple OSS projects would diminish the effects of these confounding variables and capture a more general picture of the effects of programming language diversity on growth.

Another issue is that our measure of growth looked at average growth for subsystems from versions 5 to 65, even though some subsystems weren't created until later versions, and some were removed before version 65. This means that subsystems that grew very quickly but only existed for a small subset of the versions would appear to have slow growth when averaged over every version.

5 CONCLUSION

When it comes to React Native, the motivations for using multiple programming languages are specific to the project's goals. A cross-platform framework must consider the interface demands of iOS and Android. The languages of a subsystem are then dependent on the context of the subsystem.

Although we hypothesised that more language-diverse subsystems would draw less attention from development, our analysis revealed that 10% of subsystem growth rate is due to the number of languages used. We found that core subsystems both present the most language diversity as well as motivate the growth of connected subsystems, perhaps with fewer languages. However, there exist subsystems with significant language diversity that evolve at a slower rate since they are more independent from the core.

We conclude that, in the case of React Native, highly language-diverse core subsystems naturally drive the growth of a cross-platform framework. The maintenance and development of these subsystems will be a leading factor in the growth of the software as a whole, and we can use this to infer where React Native will see future growth. Further research is needed across other OSS projects with similar language diversity in order to discover if this correlation is consistent across other projects, and how subsystem dependency influences this growth.

6 ACKNOWLEDGMENTS

We would like to thank Dr. Neil Ernst for his feedback and guidance throughout this project.

7 REFERENCES

[1] Godfrey and Qiang Tu. 2000. Evolution in open source software: a case study. *Proceedings International Conference on Software Maintenance ICSM-94* (2000). DOI:https://doi.org/10.1109/icsm.2000.883030

[2] Facebook (2015) React Native [Source code]. https://github.com/facebook/react-native.

[3] Siim Karus and Harald Gall. 2011. A study of language usage evolution in open source software. *Proceeding of the 8th working conference on Mining software repositories - MSR* (2011). DOI:https://doi.org/10.1145/1985441.1985447

[4] Tom Arbuckle. 2011. Measuring multi-language software evolution. *Proceedings of the 12th international workshop and the 7th annual ERCIM*

workshop on Principles on software evolution and software evolution - IWPSE-EVOL (2011). DOI:https://doi.org/10.1145/2024445.2024461

[5] AlDanial (2021) cloc. https://github.com/AlDanial/cloc.

[6] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The promises and perils of mining GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014* (2014). DOI:https://doi.org/10.1145/2597073.2597074

Appendix A

The data we extracted from releases of React Native can be found in the following GitHub repository: https://github.com/samuel-barrett/SEng-480B-Project

Appendix B

Samuel Barrett:
data acquisition, analysis, editing report

Oscar Sandford:
design, writing proposal and report

Natasha Bansal:
video presentation, original study analysis

Kenil Shah:
video presentation, original study analysis