Project Proposal (React Native)

Kenil Shah, Natasha Bansal, Samuel Barrett, Oscar Sandford

October 12, 2021 (last updated)

In order to understand the pains and perks of modern open source software development, we aim to recreate Tu and Godfrey's case study on Linux. Version control logs allow us to extend their 20 year old methods, employ our own scripts, and to use modern code analysis tools like CodeScene. Instead of Linux, we will make a case study of the prominent open source framework React Native. Maintained by Facebook, React Native is a 6 year old framework that is designed to develop cross-platform applications. According to GitHub, React Native consists of JavaScript, Java, C++, and more.

Tu and Godfrey's study was based on a C-focused code base, but we want to understand the effects of multiple programming languages on a project. What motivates the need for more than one language? In what subsystems are the languages used and does this lead to less manageable complexity? How much of the work is done by Facebook and how much by independent contributors, or even other companies? How has OSD changed between Linux then and React Native now? Our paper will aim to provide insight on the efficacy of using multiple programming languages in a large project, as well as the efficacy of open source for projects headed by large companies. We will also present new methods for analysing open source projects.

Now that we have our research questions and motivations down, our plan involves downloading the stable releases of React Native at roughly every quarter and modelling them in CodeScene. Samuel, who has access to CodeScene, will lead in this area. After attaining this data, two members will write analysis tools similar to the ones used in the original Linux paper. Any two members will write original scripts that enable us to grade React Native's subsystems based on file complexity (i.e. number of functions, conditionals, and import statements) and language density (i.e. which subsystems use certain languages?). We will use the raw source code and analytics generated by CodeScene to answer our questions about programming languages. Oscar has volunteered to write and edit the lion's share of the report, as well as compose a prepared video presentation.

The workflow will start from the React Native GitHub repository. All stable releases from when React Native was created are available on Github, meaning that we can study them and make educated initial predictions, run the project through CodeScene, make conclusions about our high-level questions (i.e. subsystem growth, density of programming languages, contributions over time). We will then use

scripts on stable releases every quarter to gather data about file complexity, language density, and compare these to what we see in the growth of subsystems. We expect the development of subsystems with the use of multiple programming languages to be a leading factor in React Native's complexity over time.

Potential limitations of this study chiefly lie in our focus on quantitative data from a single project on GitHub. This is a case study, and our analysis will not be indicative of the effects of programming languages on all large OSD projects. Furthermore, we are less knowledgeable undergraduates, therefore our qualitative conclusions on the data will be skewed by our lack of experience as developers and scientists. Daniel German's paper on the "perils and promises" of mining Git and GitHub will be an aid to mitigating misjudgments on our data.