Linux Kernel Best Practices and Project Rubric

Aditya Khadse
akkhadse@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Anmol Lunavat
alunava@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Deep Mehta
dmmehta2@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Pritesh Surana
psurana@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Rohit Nair
rsnair@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

This document states the connection between the Linux Kernel Best Practices and the Project Rubric provided as part of the Software Engineering coursework. This is based on personal understanding and opinion of the Group 5 project team.

KEYWORDS

linux kernel, software sustainability, rubric

ACM Reference Format:
Aditya Khadse, Anmol Lunavat, Deep Mehta, Pritesh Surana, and Rohit Nair. 2022. Linux Kernel Best Practices and Project Rubric. In *Proceedings of Software Engineering - CSC 510 Fall 2022 (Conference acronym 'SE')*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Linux Kernel Best Practices are lessons learnt by the kernel project team over the course of 26 years of developing and actively maintaining the Linux Kernel project. These lessons have been studied to understand the reason behind the success of this open source project.

The Project Rubric provided as part of the Software Engineering coursework is a detailed rubric to ascertain that a particular repository is a good open source repository and it meets all the expected standards. These rubric metrics also include the Software Sustainability Evaluation metrics that have certain metrics defined to ensure that the project is designed and developed in such a way that it has a better reach to the users and developers of the open source project.

In this paper we will try to list down the connection between the Linux Kernel best practices and the Project Rubric.

Conference acronym 'SE', Oct 6, 2022, Raleigh, NC

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 COMMON METRICS

As noted in the introduction, we list down the commonalities between the Linux Kernel practices and project rubric and try to understand the importance of each of them.

2.1 Short Release Cycles

The Linux Kernel case study shows how in the past they had long release cycles of couple of years. This meant that users had to wait a long time before new features were available or before an old bug was resolved. In addition to this, it wasn't very developer friendly, since developers rushed towards the end of release cycles to push their part of code in current cycle, because they knew if they missed the current deadline they would have to wait a long time before their contributions were deployed in next release.

Thus they benefited a lot by implementing shorter release cycles, usually couple of months. Following benefits were noted by Linux Kernel team:

- New and updated code was quickly made available to the end users.
- Developers focus on quality without worrying about pushing their code in release cycles.

Similar points related to short release cycles can be found in the project rubric:

- Short release cycles.
- Are releases tagged in the repository?.

2.2 Distributed Responsibilities for Scalability

In the beginning, all the Linux Kernel pull request reviews were sent to one person and was his responsibility to ensure the code was stable and new code didn't break anything. But with time and growth, they learnt that spreading over the responsibilities of code review across multiple users increased efficiency without compromising quality.

Similarly in the project rubric it specifies the following:

- Workload is spread across the whole team.
- Chat channel exists between the widespread team.

2.3 Use of right tools

Linux Kernel case study shows how their transition to tools like Bitkeeper source code management and git proved effective in its growth. Without these tools the Kernel wouldn't be where it

is today. Thus selecting the right set of tools is as important as anything else for the success of the project.

The project rubric also ensures that the right and consistent set of tools are used across the project team. In addition to this it also ensures automated tools are in place to ensure the new changes being merged are up to the standard and they don't break anything. In addition to this, it also checks that the tools that are used are documented so that anyone who wants to contribute to the project has an idea about the ideal set of tools they need to install as a pre-requisite.

2.4 Consensus-Oriented model

Linux Kernel practices mentions that every users opinion matters and no user or group of users is given priority over others. This ensures that the kernel remains usable and suitable to a whole range of users and not just a particular section of society.

In connection to this practice the project rubric has following checks:

- Do you have a governance model?
- Do you have a contributions policy?

These checks help to make sure that there are policies defined publicly for the contributors.

2.5 Strong No Regressions Rule

Linux Kernel practices mentions that the newer versions and updates made to the system ensure that everything that worked with older versions work as well. This is really important for users to update confidently and not having to worry that the new updates will break their existing work.

The project rubric thus checks if tests are in place with good code coverage to ensure that everything works as expected. In addition to this, it checks for automated checks with run as nightly builds to verify that the system is stable.

2.6 No single entity controls the development

Linux Kernel practices mention that while anyone can improvise and contribute to Linux kernel as per their requirements, no one entity can push it in a direction that is beneficial for them and at the same time harmful to some other entity.

In connection to this practice the project rubric has following checks:

- Does each of your source code files include a copyright statement?
- Is your software released under an open source licence?

Thus the licence, copyright and a consensus oriented model helps ensure a level playing ground for all the developers and users of the system.

2.7 Anyone can contribute anywhere

Linux Kernel best practices eliminates any type of bureaucracy and instead allows developers to contribute on any part of the code. This allows quicker resolution of bugs and also provides developers a better understanding of overall project rather than just a part of it.

The contribution guidelines check in project metric does exactly this to ensure that the contributors are well aware of their responsibilities and rights.

3 CITATIONS AND BIBLIOGRAPHIES

- 2017 Linux Kernel Development Report - Jonathan Corbet, LWN.net Greg Kroah-Hartman, The Linux Foundation
- Software Sustainability Evaluation - Software Sustainability Institute

Received 7 October 2022