Slash rubrics mapping to Linux Kernel Practices

Shubham Mankar – themrshubh@gmail.com
Rahil Sarvaiya – rahil1304@gmail.com
Moksh Jain – mokshjain98@gmail.com
Pratik Devnani – pratikdevnani24@gmail.com
Anushi Keswani – keswanianushi@gmail.com

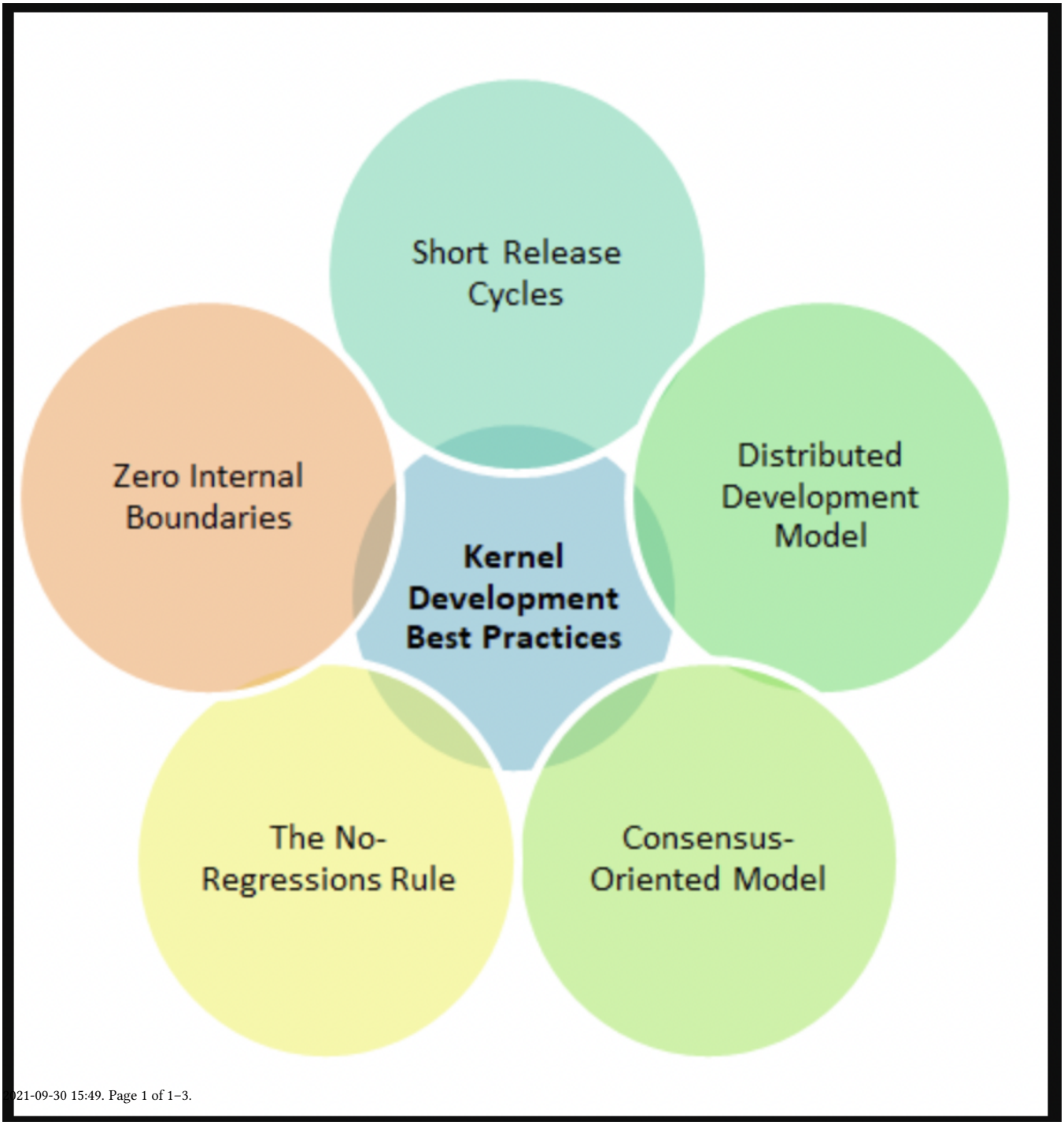

Figure 1: Linux Kernel Practices

ACM Reference Format:
Shubham Mankar – themrshubh@gmail.com, Rahil Sarvaiya – rahil1304@gmail.com, Moksh Jain – mokshjain98@gmail.com, Pratik Devnani – pratikdevnani24@gmail.com, and Anushi Keswani – keswanianushi@gmail.com. 2021. Slash rubrics mapping to Linux Kernel Practices. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SHORT RELEASE CYCLES

Shorter release cycles effectively ensures less frustration for the user and the developer alike. It reduces the chances of merging and pushing inefficient and unstable code as shorter cycles ensure regular testing and up-date knowledge of the system architecture. This method also helps to bring fast and positive changes which result in the end user being satisfied. For these reasons, we integrate new code in short cycles.

The following rubrics can be mapped to this practice

Rubric	Evidence
Number of commits	https://github.com/secheaper/slash/pulse
Number of commits: by different people	https://github.com/secheaper/slash/pulse
Issues reports: there are many	GH Issues
Issues are being closed	GH closed issues
Tests that can be run after your software has been built or deployed to show whether the build or deployment has been successful	GH actions
Automated test suite for your software	GH actions
Framework to periodically (e.g. nightly) run your tests on the latest version of the source code	GH actions
Using continuous integration, automatically running tests whenever changes are made to your source code	GH actions
Test cases are routinely executed	GH actions

2 DISTRIBUTED DEVELOPMENT MODEL

A distributed Development model is the best way to develop any software. Sharing different functionalities of the software to different individuals, based on their familiarity with the area ensures seamless code review and integration with very minimal chances of blow-up. For this reason, Distributed Development Model has been followed.

The following rubrics can be mapped to this practice

Rubric	Evidence
workload is spread over the whole team	GH commits
evidence that the whole team is using the same tools	GH
E-mails to our support e-mail address are received by more than one person	communication tab
Listing the important partners and collaborators on our website	GH
Do we accept contributions from people who are not part of your project?	GH
Do you have a contributions policy	GH
Is your contributions' policy publicly available?	GH
Evidence that the members of the team are working across multiple places in the code base	GH

3 CONSENSUS-ORIENTED MODEL

Integration to the code base need to be agreed upon by all and especially by people who have implemented some functionality and the new code block directly works with that. This ensures not tampering with the stable versions of code.

The following rubrics can be mapped to this practice

Rubric	Evidence
Chat channel: exists	https://discord.com/channels/879343473940107264
issues are discussed before they are closed	every issue is discussed by all, then assigned to one appropriate person for closure

4 THE NO-REGRESSIONS RULE

The No-regression rule is an important design decision as once the interface with the model gets pushed and is in public use, we should not alter that syntax. This ensures harmony in terms of user calls and less frustrations. We have ensured that we don't take away existing functionality but add to it.

Rubric	Evidence
Use of version control tools	Git is used thoroughly through the project
Evidence that the members of the team are working across multiple places in the code base	https://github.com/secheaper/slash/graphs/contrib
There is a branch of the repository that is always stable	the main branch is always stable

5 ZERO INTERNAL BOUNDARIES

We understand that access to the entire view of the project is important. Even though individuals are working on different functionalities, it does not stop them from making changes in other parts of the code. This results in problems being solved at the source rather than having multiple paths to go through before making actual changes.

The following rubrics can be mapped to this practice

Rubric	Evidence
whole team is using the same tools	We can clearly see that entire codebase has been written in Python(https://github.com/secheaper/slash/search?l=python). Everyone has the same access to the repository and also have equal access to comitting directly to the main branch.
issues are discussed before they are closed	There is a discussion channel on our discord server
Source code publicly available to download, either as a downloadable bundle or via access to a source code repository	git clone https://github.com/secheaper/slash.git or Download as a zip file from here https://github.com/secheaper/slash/archive/refs/heads/main.zip
E-mails to our support e-mail address are received by more than one person	we all have the access credentials to the support email
Project have a ticketing system to manage bug reports and feature requests	We constantly create issues and have an ticketing system on github projects where we assign each member an issue based on priority. (https://github.com/secheaper/slash/projects/1)