Auction Sphere

Kartik Soni
ksoni@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Palash Rathod
prathod@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Shreya Maheshwari
smahesh4@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Nandini Mundra
nmundra@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Tanvi Sinha
tsinha2@ncsu.edu
North Carolina State University
Raleigh, NC, USA

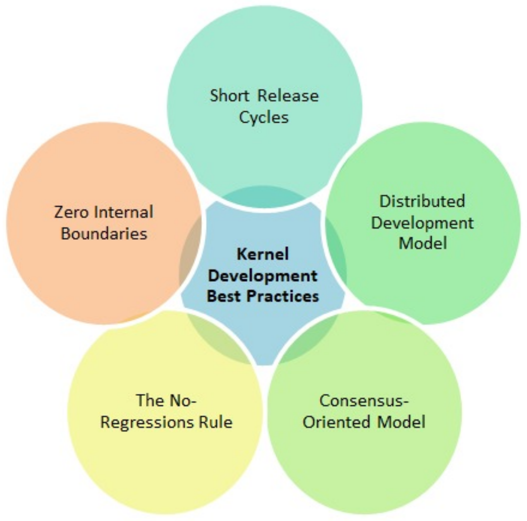

Figure 1: Linux Kernel Best Development practices

ABSTRACT

Auction-Sphere is an easy to use application that allows user to buy and sell products online. It is an online platform to bid for the product one wants to sell. People are allowed to create their profiles, post products, make multiple bids and get an overall insight of the top 10 bids. This document is way to understand the Linux Kernel best practices and implement the same while completing a new development project. We applied those practices in our project and ultimately, we were able to connect the project rubrics to all of the Linux Kernel best practices.

KEYWORDS

Auction System, Bidding, Software Engineering

ACM Reference Format:
Kartik Soni, Palash Rathod, Shreya Maheshwari, Nandini Mundra, and Tanvi Sinha. 2018. Auction Sphere. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 DISTRIBUTED DEVELOPMENT MODEL

Following a Distributed Development Model ensures that all the members of the team are actively contributing to the code base and that the responsibility is not confined to just a single person. It also allows all the members of the team to have an idea of the different features that are being implemented. In this project, Distributed Development Model was followed in the following ways: 1) GitHub was used as the version control system. 2) Work was divided amongst the team members according to their experiences and proficiency. 3) Issues were created for everyone to work on close. 4) The issues were closed after a discussion with other team members. The evidence for this can be found in the 'Insights' tab of the GitHub repository where the closed issues can be viewed.

Along with that, the commits done by the members can be viewed indicating that all the members have contributed to the code base.

2 SHORT RELEASE CYCLES

Earlier, major releases were done once in few years which caused delays in getting new features to customers and also resulted in large pieces of code being merged together to the old code. Short release cycles ensure that enhancements and bug fixes are incorporated regularly resulting in no or very minimal integration issues. Short release cycles are measured in the rubric as well. Version control tools help make short release cycles possible which is measured in the rubric point- "`Short release cycles`"

3 THE NO-REGRESSIONS RULE

Regression testing is done to avoid changing existing functionality when there are changes made to the existing code-base or a new feature is added. This assures users there won't be any change in the existing functionality of the application.

The following measures are in place to ensure this holds:

- Usage of version control tools
- Test cases exist and cover more than 30% of the code base
- Test cases are executed when new changes are pushed to the code base

4 ZERO INTERNAL BOUNDARIES

All 5 people on our team are using the same tools for development - React.js, Flask, and SQLite, and everyone has access to these tools. Further, we have one public repository with all the code of our project including frontend and backend - and although some contributors have more contributions in the React frontend, some more in the backend, everyone has access to changing both parts of the project as required. All contributors are able to install dependencies and run the React application as well as the Flask server. This practice is enabling easier debugging of issues. For instance, if an API call fails at the frontend, the frontend contributor can look at the specific backend endpoint and figure out what the problem could've been right there and then.

5 CONSENSUS ORIENTED MODEL

Consensus Oriented Model means that any changes to the project must be approved by majority contributors of the project. This helps in making sure that none of the code base is jeopardized when a new update is made. It also helps in verifying correctness of the code that is being released. One of the best practices is to schedule meetings to address open issues during the project development. This discussion ensures that the solution to the problem is acknowledged by the other developers and the code does not conflict with any previously written code. Even after pushing the new changes to GitHub, many tests are rigorously conducted to make sure the code is working fine. Following are the rubrics associated to these practices:

- Issue reports: there are many.
- Issues are being closed.
- Issues are discussed before they are closed.