

# 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

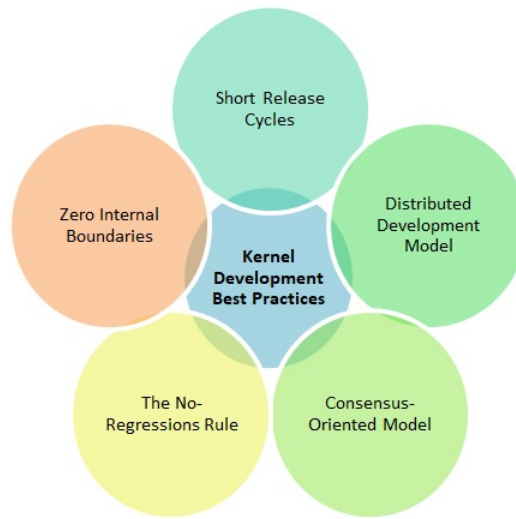


Figure 1: Linux Kernel Best Development practices

## ABSTRACT

## 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>

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or commercial use is granted by ACM, Inc. provided that the fee of \$15.00 is paid directly to ACM. This permission is granted without fee or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
Conference'17, July 2017, Washington, DC, USA  
© 2018 Association for Computing Machinery.  
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00  
<https://doi.org/XXXXXXX.XXXXXXX>

2022-10-09 17:51. Page 1 of 1-2.

## 1 DISTRIBUTED DEVELOPMENT MODEL

## 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.

Unpublished working draft.  
Not for distribution.