Devops HW-1

Name : Chadaram Geyanth Padmakar

Roll.No : 205224002

Branch: M.Tech Data Analytics

Department: Department of Computer Application

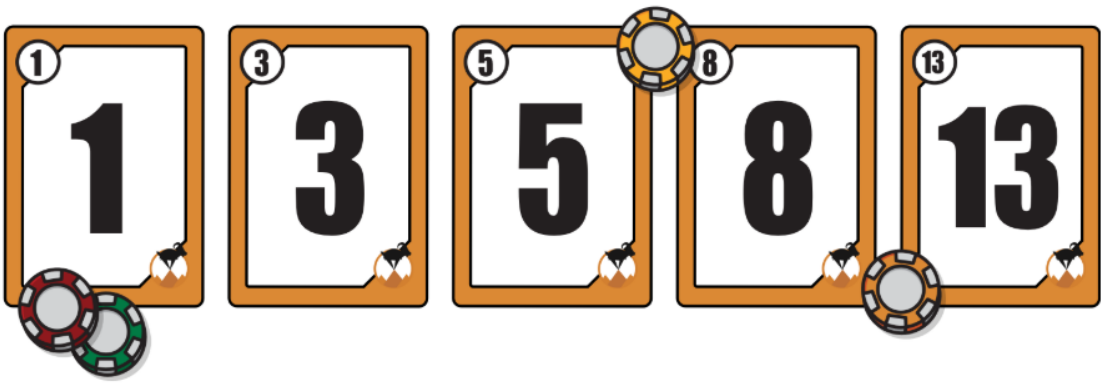
Q1: Read about "Planning Poker" - Agile estimation technique and illustrate an example with a Development Team of 10 who are tasked to develop a mobile app for Maha-Khumb in 3 months.

Introduction to Planning Poker

Planning Poker is an agile estimation technique used by development teams to estimate the effort required for user stories or tasks. It helps teams make collaborative, unbiased, and accurate estimates in a fun and engaging way.

How Planning Poker Works

1. **Select a Moderator** – Usually, the Scrum Master or Product Owner facilitates the session.
2. **Prepare User Stories** – The Product Owner presents user stories that need estimation.
3. **Distribute Poker Cards** – Each developer receives a deck of cards with Fibonacci sequence values (1, 2, 3, 5, 8, 13, 21, etc.).



4. **Discuss the Story** – The team clarifies doubts and understands the complexity.
5. **Vote Simultaneously** – Team members select a card representing their estimated effort and reveal it at the same time.
6. **Discuss Differences** – If estimates vary significantly, the team discusses reasons and re-votes.
7. **Finalize the Estimate** – Once a consensus is reached, the final estimate is recorded.

Using Planning Poker for Estimating a Maha-Khumb Mobile App Development

A development team of **10 members** is tasked with building a **mobile app for Maha-Khumb** within **3 months**. The app needs to support key features like **user registration, event schedules, GPS navigation, push notifications, and vendor listings**. The team decides to use **Planning Poker** for estimation.

Step 1. Introduction to Estimating Approaches

Before starting, the **Scrum Master (facilitator)** explains the estimation process. The team agrees to use **Planning Poker**, based on the **modified Fibonacci sequence** (1, 2, 3, 5, 8, 13, etc.) for estimating effort in story points.

Step 2: Understanding Planning Poker

- Each team member receives a **deck of Planning Poker cards**.
- The **Product Owner** presents a **user story** from the backlog.
- Example **User Story**:
  "As a user, I want to register and log in using my phone number and OTP."
- The team **discusses the complexity**, including authentication, OTP verification, and database storage.
- After the discussion, all members **simultaneously reveal their cards**.

Step 3: Consensus Building in Estimation

First Round of Estimation

- Team members reveal different estimates: **3, 8, 2, and 5**.
- Since estimates vary, the **Scrum Master asks outliers** (e.g., the one who picked 8 and the one who picked 2) to explain their reasoning.
    - The developer who picked **8** thinks **OTP security and backend authentication** will take more effort.
    - The developer who picked **2** believes **using Firebase authentication** will reduce complexity.

Step 4: Iterative Estimation Process

- After discussion, team members **re-estimate** and reveal their cards again.
- New estimates: **5, 5, 5, 8, 5, 5, 5, 5, 5, 5** → Majority agrees on **5 story points**, so it is recorded.

Step 5: Estimating Other Features

The same process is repeated for other **user stories**:

User Story	First Estimates	Final Estimate
GPS Navigation	8, 13, 8, 8, 5	**8 Story Points**
Live Event Updates	5, 5, 3, 5, 5	**5 Story Points**
Push Notifications	3, 3, 3, 3, 2	**3 Story Points**
Vendor Listings	8, 5, 5, 5, 5	**5 Story Points**
Emergency Contacts	2, 3, 2, 3, 2	**2 Story Points**

Step 6: Handling Discrepancies in Estimates

- **Scenario 1**: The **GPS Navigation feature** had a **split between 8 and 13**.
    - Some developers argued it was complex due to **real-time tracking**, while others felt using **Google Maps API** would simplify it.
    - The **team decided on 8 story points** after agreeing on the best technical approach.
- **Scenario 2**: The **Push Notifications feature** had **3 and 2** as estimates.
    - The **developer who picked 2** thought it was simple with **Firebase Cloud Messaging (FCM)**.
    - The **developer who picked 3** considered additional effort for **custom notification templates**.
    - The team settled on **3 story points** after discussion.

Step 7: Establishing Baselines for Estimation

- The team **selected the "User Registration" story (5 points) as a baseline**.
- They then identified another **"Live Event Updates" story as a 5** to **validate consistency**.
- This **helped ensure all future estimates remain relative** to these baselines.

Step 8: Using Planning Poker Resources

- The **Scrum Master** suggested using an **online Planning Poker tool** for remote team members.
- The team is now **aligned with estimates** and can move forward with **Sprint Planning**.

Conclusion

- Using **Planning Poker**, the **Maha-Khumb mobile app development team** successfully estimated effort, resolved uncertainties, and built a **realistic development plan** for completing the app within **3 months**.

Q2. Read Paper – Measuring Software Development Waste in OSS Projects - https://arxiv.org/pdf/2409.19107. Pick one measure from this paper and apply it on any open-source repository. Share results.

Implementation of PR Rejection Rate from the Research Paper

Selected Measure: PR Rejection Rate

From the research paper, one of the key software development waste metrics is **Pull Request (PR) Rejection Rate**. This metric helps in evaluating the efficiency of the software development process in open-source projects.

Formula for PR Rejection Rate:

PR Rejection Rate = Unmerged Closed PRs/Total Closed PRs

Where:

- **Unmerged Closed PRs**: The number of PRs that were closed but not merged.
- **Total Closed PRs**: The sum of merged and unmerged closed PRs.

A high rejection rate may indicate inefficiencies, such as misalignment between contributors and maintainers, poor code quality, or redundancy in contributions.

Application on an Open-Source Repository

For this implementation, I analysed the **Next.js repository** (https://github.com/vercel/next.js) using the GitHub API. The repository is one of the most popular React frameworks for server-side rendering and static site generation.

Step 1: Fetch PR Data

Using a Python script, I retrieved PR data from the Next.js GitHub repository and calculated the PR rejection rate.

Step 2: Results from Next.js Repository

```
D:\M.Tech\2nd Sem\Devops\Assignments\hw1-effort-estimation-and-software-waste-Geyanth08>python script.py
Total Closed PRs: 100
Merged PRs: 84
Unmerged PRs: 16
PR Rejection Rate: 0.16
```

Interpretation:

- **84% of closed PRs were merged**, indicating a high acceptance rate.

- **16% of PRs were closed without merging**, meaning some contributions were either rejected or abandoned.
- A **low rejection rate (0.16)** suggests that the project maintainers actively engage with contributors, and most PRs meet the required standards.

Conclusions

The **Next.js project has an efficient PR process** with a high acceptance rate.
A **PR rejection rate of 16% is relatively low**, meaning that most contributions add value.