FreeRTOS Architecture Part 1

Name

Universidad Panamericana

Presentation August 5, 2024



Contents

- Why good practices?
- Use other programming languages
- Requirements
- SDD
- **5** UML
- Testing
- TPD

Why good practices?



Why good practices?

1. Maintainability

 This is crucial for long-term projects where multiple developers might be working on the same codebase over time.

2. Readability

 Clear and consistent coding standards make it easier for developers to read and understand each other's code.

3. Reusability

 This means that code can be reused in different parts of a project or even in different projects, saving time and effort.

4. Bug Reduction

Identifying and fixing bugs early in the development process.

5. Performance

 This is particularly important in applications where performance is critical, such as real-time systems or high-traffic web services.

Why Good Practices?

6. Scalability

Be easily extended with new features wiYout significant rework.

7. Security

Secure data handling are crucial in preventing security breaches.

8. Documentation

• For future maintenance, debugging, and onboarding new developers.

9. Consistency

 It allows developers to switch between different parts of the codebase wiYout needing to adjust to different coding styles.

10. Professionalism

 It can enhance the reputation of a development team or company and build trust with clients and stakeholders

10 commandments

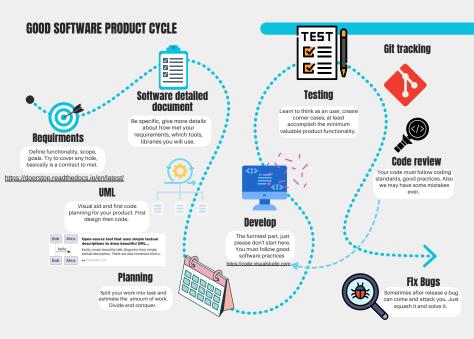
- You shall prioritize Maintainability
- You shall value Readability
- You shall strive for Reusability
- You shall reduce Bugs early
- You shall optimize Performance
- You shall ensure Scalability
- You shall secure thy code
- You shall document thoroughly
- You shall maintain Consistency
- You shall uphold Professionalism

Use other programming languages



You shouldn't be opposed to learning programming languages.

- Latex : Documentation
- Java: PlantUML
- JSON: For automatization stuff
- Python: Unit Testing
- C: Pretty basic programming
- Cpp: Robust programming
- C#: Windows App



Requirements



Requirements

DOORS is a requirements management tool used by organizations to manage project requirements throughout the development lifecycle. It helps in capturing, analyzing, tracing, and maintaining changes to information to ensure a project's compliance with its initial requirements. This tool is particularly useful for managing complex projects, providing traceability, and improving collaboration and verification efforts. Requirements

SDD



Software Detailed Document (SDD)

The SDD serves as a guide for developers during the build phase and aids in maintaining consistency and understanding of the system's design principles and functionalities. SDD



UML

UML



Unified Modeling Language with PlantUML (UML)

UML diagrams serve as excellent documentation tools that are useful throughout the system's lifecycle. They help new team members understand the system quickly and can also be valuable for maintenance and future upgrades. PlantUML

Testing



Black box vs Whitebox testing

- Black Box Testing: Tests the functionality of software without knowledge of its internal workings, focusing on input and output.
- White Box Testing: Examines the internal structure and workings of software, requiring knowledge of the code to ensure through testing of internal operations.

TPD



Test plan document (TPD)

A Test Plan Document (TPD) outlines the strategy, resources, scope, and timeline for testing activities within a software project. It serves as a blueprint that guides the testing process, detailing what needs to be tested, how the testing will be conducted, who will perform the tests, and the expected outcomes TPD