Conventional or TDD Approach

Project Training –Automotive Overview

24/07/2024
Areeb Hammad N
Anuratha N
Atraindra Gupta
Anji Babu

Version: 1.0

Created: 24/07/2024

Last Updated: 24/07/2024

Status: DRAFT (The status would change to finalized post the BA, PM and dev team review and sign off)

Document History- To maintain a list of changes being made

Version	Date	Author	Description of change
1	24/07/2024	Areeb Hammad N	Draft
2	24/07/2024	Anuratha N	Draft
3	24/07/2024	Atraindra Gupta	Draft
4	24/07/2024	Anji Babu	Draft

Approvers List- To track who has reviewed and signoff on the Test plan

Name	Role	Approver/Reviewer	Approval/Review Date
Abhishek	Professor	Reviewer	

Reference Documents-Clearly mark the document used as an input to create the test plan

Version	Date	Document Name

Task1: which testing algorithm you will choose whether conventional or TDD for calculator and smartphone products and why? 2 pointers are sufficient

Calculator

Conventional Testing:

Calculators are relatively simple devices with well-defined functionalities (e.g., addition, subtraction, multiplication, division). Conventional testing can be efficient here because:

1. **Predefined Test Cases:** You can create comprehensive test cases covering all possible operations and edge cases.
2. **Less Frequent Changes:** Calculators typically don't undergo frequent updates or changes, making conventional testing sufficient

Why Not TDD for Calculator?

1. **Simplicity**: Calculators have relatively simple functionalities that are straightforward to test with predefined test cases. The operations (addition, subtraction, multiplication, division) are well-defined and less prone to frequent changes.
2. **Overhead**: TDD involves writing tests before the actual code, which can be seen as an unnecessary overhead for such a simple device with limited functionalities.

Smartphone

TDD:

For software development on smartphones (e.g., apps, OS features), TDD is highly advantageous because:

1. **Frequent Updates:** Smartphone software is frequently updated, and TDD ensures that new changes don't break existing functionality.
2. **High Complexity:** The complexity of smartphone software, with numerous features and integrations, benefits from the rigorous testing and refactoring cycles of TDD.
3. **Improved Collaboration:** TDD facilitates better collaboration among developers by providing clear, testable requirements.

Why Not Conventional Testing for Smartphone Software?

1. **Frequent Updates**: Smartphone software is frequently updated with new features, bug fixes, and improvements. Conventional testing might not catch all issues in such a dynamic environment, leading to potential regressions.
2. **High Complexity**: Smartphone applications and OS features are highly complex, involving multiple integrations and dependencies. Conventional testing may not be sufficient to ensure comprehensive coverage and robustness.