Release notes for TDD implementing cruise control

Name of Testers:
1. Anupam
2. Asha
3. Harshvardhan
4. Hrushikesh
5. Manish
6. Krishna
7. Tahir

Version: 1.7
Created: 05/28/2024
Last Updated: 05/28/2024
Status: Complete

Table of Contents

1. Introduction

This document provides detailed release notes for the implementation of the cruise control feature in a car using Test-Driven Development (TDD) and AUTOSAR standards. The cruise control feature allows drivers to set and maintain a desired speed automatically, enhancing the driving experience and reducing driver fatigue. This implementation adheres to industry best practices to ensure reliability, performance, and user satisfaction.

2. Feature Overview

The cruise control system is designed to enable a vehicle to maintain a specified speed without the driver needing to apply constant pressure on the accelerator. Key functionalities include:

- Speed Setting: Allows the driver to set a desired speed.

- Speed Adjustment: Enables the driver to increase or decrease the set speed.

- Speed Maintenance: Ensures the vehicle maintains the set speed under various driving conditions.

- Disengagement: Allows the driver to easily disengage the system, typically through braking or manual switch-off.

This implementation follows the AUTOSAR (AUTomotive Open System ARchitecture) framework, which ensures a standardized and scalable automotive software architecture. AUTOSAR compliance guarantees that the system can integrate seamlessly with other automotive systems and supports modularity for future enhancements.

3. Test-Driven Development (TDD) Process

The TDD process is integral to ensuring that the cruise control feature is developed with high quality and minimal defects. TDD involves writing tests before writing the actual code, which ensures that the code meets predefined requirements from the outset. This section provides a detailed explanation of each step in the TDD process as applied to the cruise control feature.

Step-by-Step TDD Process

Step 1: Start with a Feature Request

- Description: Identify the need for the cruise control feature based on user requirements and project goals. This involves gathering detailed requirements and understanding the functionality from the end-user's perspective.

- Example: A feature request might specify that the cruise control system must allow the driver to set and maintain a speed between 30 and 120 km/h, with the ability to adjust the speed in 1 km/h increments.

Step 2: Write a Test

- Description: Create a test case that outlines the expected behavior of the cruise control feature. This test case acts as a specification and ensures that the development starts with a clear understanding of the desired outcome.

- Example: Write a test to verify that when the driver sets the speed to 60 km/h, the vehicle maintains that speed. Another test might check if the system correctly adjusts the speed when the driver increments or decrements it.

Step 3: Test Fail

- Description: Run the test to ensure it fails initially. This step confirms that the test case is correctly designed to detect the absence of the feature. A failing test indicates that the feature is not yet implemented.

- Example: Execute the test suite and observe that the test for setting the speed to 60 km/h fails because the cruise control functionality is not yet present in the system.

Step 4: Write Code

- Description: Develop the code necessary to implement the cruise control feature. The initial focus is on writing just enough code to pass the failing test. This ensures that development is driven by the test requirements.

- Example: Implement the basic functionality to set and maintain a specified speed. For instance, write code that allows the system to accept a speed input and regulate the throttle to maintain that speed.

Step 5: All Tests Pass

- Description: Run the test case again to ensure it passes. This step confirms that the new code meets the requirements specified in the test. Passing tests indicate that the feature works as expected.

- Example: Re-run the test suite and verify that the vehicle now maintains a speed of 60 km/h as set by the driver. Additionally, ensure that tests for speed adjustments also pass.

Step 6: Refactor

- Description: Refactor the code to improve its structure, readability, and maintainability without changing its behavior. This step ensures that the codebase remains clean and efficient, making future enhancements easier to implement.

- Example: Simplify the logic used to maintain the set speed or improve the method of reading speed inputs. Ensure that the refactored code passes all existing tests.

Step 7: Iterate

- Description: Repeat the TDD process for additional features or improvements. Each iteration starts with a new test, ensuring continuous integration of new functionality with minimal defects.

- Example: Write new tests for additional features such as automatic disengagement when braking, or for maintaining the set speed on uphill and downhill gradients. Implement and refactor as needed, ensuring all tests pass.

By following this detailed TDD process, the development team ensures that the cruise control feature is built incrementally with a strong emphasis on quality and reliability. Each step of the process is designed to catch defects early and produce a robust, maintainable codebase that adheres to AUTOSAR standards.

THANK YOU