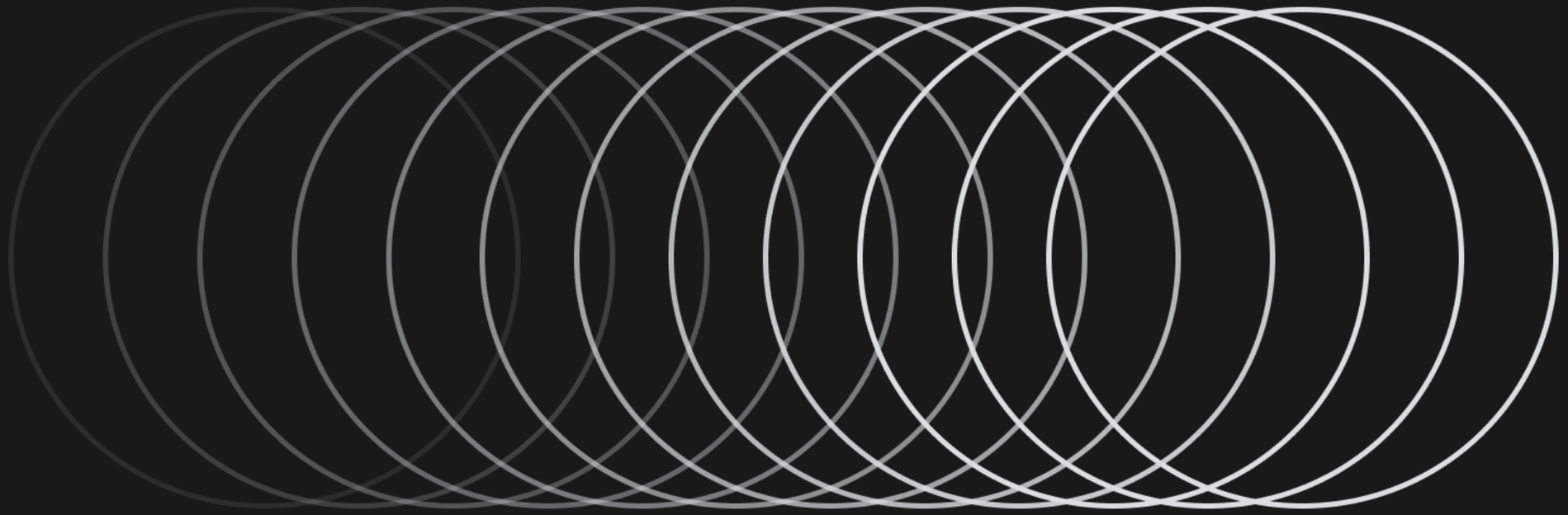
Effective Software FMEA Training

Learn a proven, unique methodology that works seamlessly in an Agile environment for mission-critical systems.



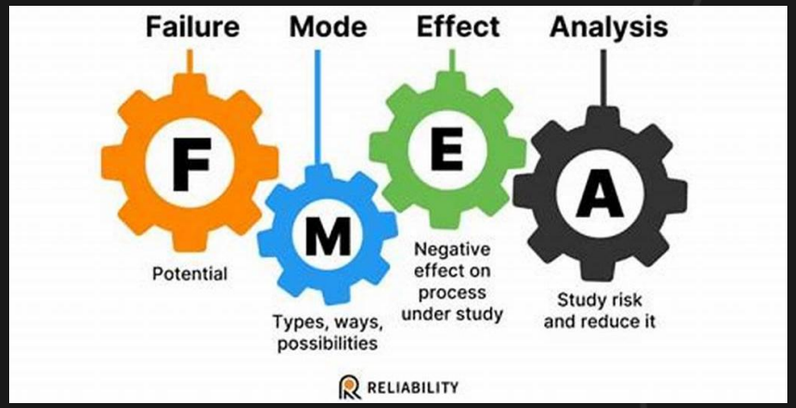
MISSION READY SOFTWARE

Contents

Software FMEA Core Value

Proven Methodology

Learn a unique methodology that works seamlessly in Agile environments, finding the most critical failure modes with minimal time and effort investment.



Time Efficient

Identify overlooked critical failure modes in days instead of months.

Agile Compatible

This approach works within any FMEA standard while maintaining agile development principles and rapid delivery schedules.

Why Traditional FMEA Fails

Avoid Common Inefficiencies

Traditional FMEA approaches expose why common methods are ineffective for software, leading to wasted time and missed risks.

Line-by-Line Analysis

- Inefficiently focuses on code implementation details
- Failures actually originate in design phase
- Wastes significant time without finding root causes

Hardware-Centric Approach

- Black box analysis fails for software
- Doesn't account for functional software complexity
- Misses critical software-specific failure modes

Requirement Analysis

Analyzing every requirement individually overlooks critical interaction failures.

Days Not Months

Our method finds critical modes in days versus traditional approaches.

Standard Compatible

Works within any FMEA standard while maintaining effectiveness.

Course Structure Overview

4
Core Modules

100
Common Defects

Free
Module Available

Days
Results Timeline

Module 1

> End-to-end short example demonstrating the complete software FMEA process in action.

- Free introductory module for all participants
- Complete walkthrough of methodology application
- Practical example using smart device analysis

Module 2

> The Most Common Software Defects covering comprehensive CDE enumeration and classification.

- Top 100 most common software defects
- Detailed Common Defect Enumerations reference
- Industry-standard defect classification system

Modules 3-4

> Advanced techniques for tagging CDEs to effects and implementing design improvements.

- Tagging CDEs to effects and hazards
- Affecting design and test case development
- Practical implementation and closure tracking

Software FMEA Process Flow – Using Smart Device Example

Analyze Smart Device

Identify what can go wrong with smart device functionality and system interactions.

Identify Relevant CDEs

Brainstorm the most common and serious Common Defect Enumerations for analysis.

Tag to Hazards

Connect system effects to PHA, FHA, or FDSC hazard analysis frameworks.

Mitigate and Track

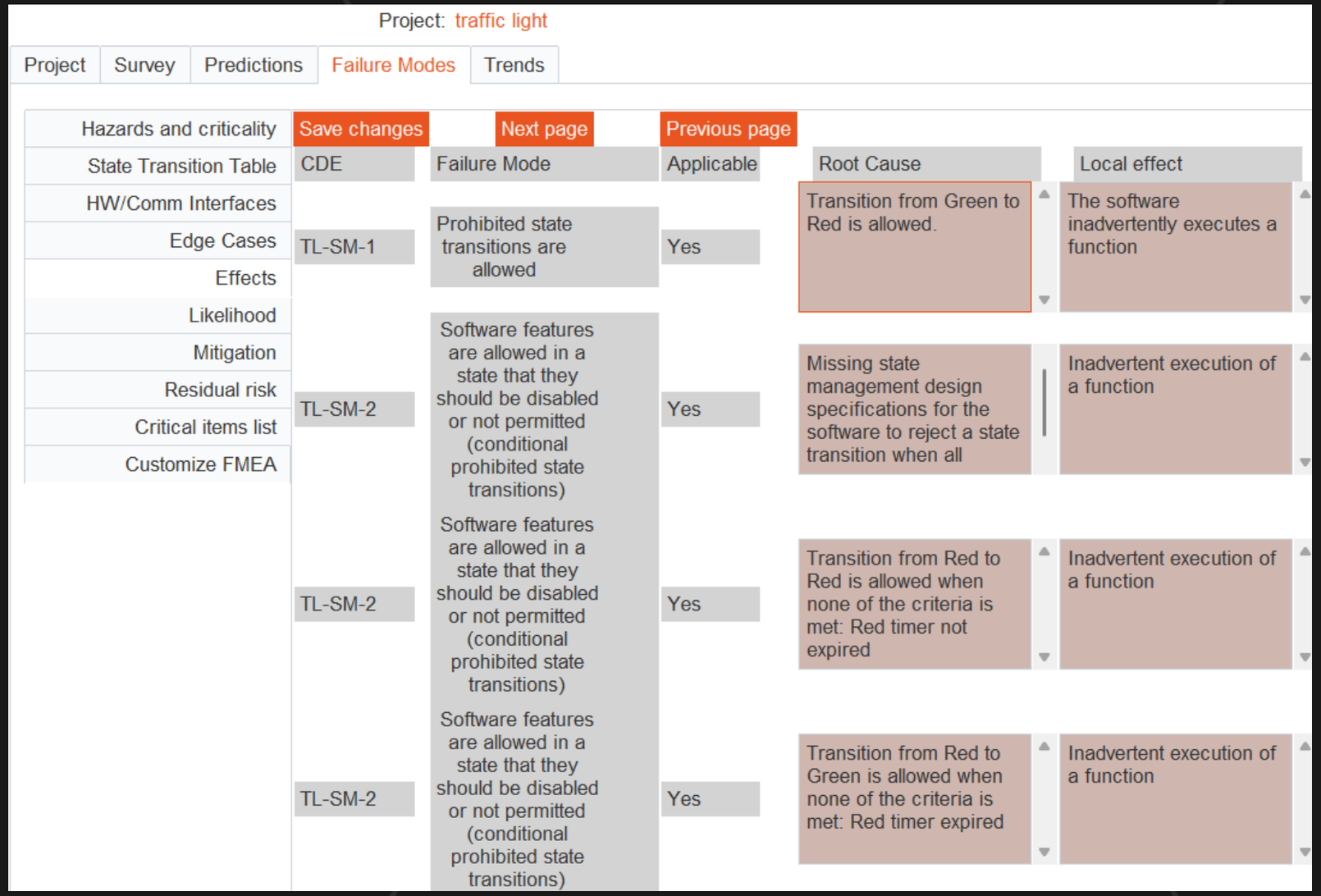
Implement controls, assess detectability, measure likelihood, and track to closure.

Identify Failure Modes

Deciding where to start is the hardest part of the software FMEA

Prohibited state transitions are one of the most overlooked edge cases

But easy to identify when you are looking for it.



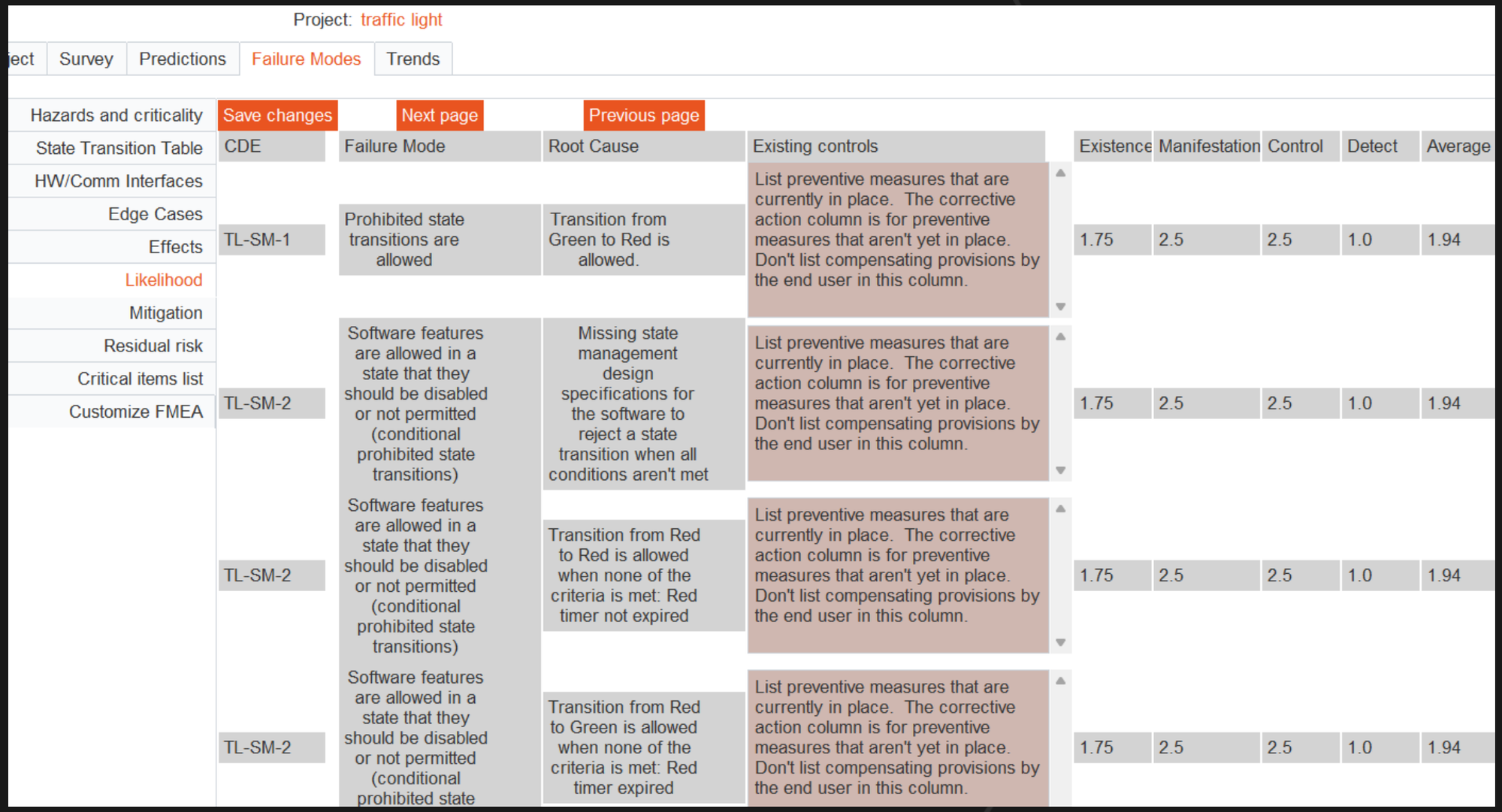
Tag effects to the FDSC, PHA, FHA

Once the root cause and local effect are determined, identify the system effect and tag it to the FDSC, PHA, or FHA, which determines its severity.

Hazard code	Description	Severity level
H1	Collision - Opposing lights are both green	1 - Catastrophic
H2	Traffic stall - no cars move when they should	2 - Serious
H3	Lights not in sync with other lights	2 - Serious

	Applicable	Root Cause	Local effect	System effect	Mission effect	Company effect	End user effect	Safety effect	Hazard	Recom
in a ey abled tted al tate	Yes	Transition from Red to Green is allowed when none of the criteria is met: Red timer expired	Inadvertent execution of a function	Collisions	Mission objectives not met	Lawsuites	Injury	Yes	H1	1.0

Assess likelihood based on objective measures



Project: traffic light

ject | Survey | Predictions | Failure Modes | Trends

Hazards and criticality
State Transition Table
HW/Comm Interfaces
Edge Cases
Effects
Likelihood
Mitigation
Residual risk
Critical items list
Customize FMEA

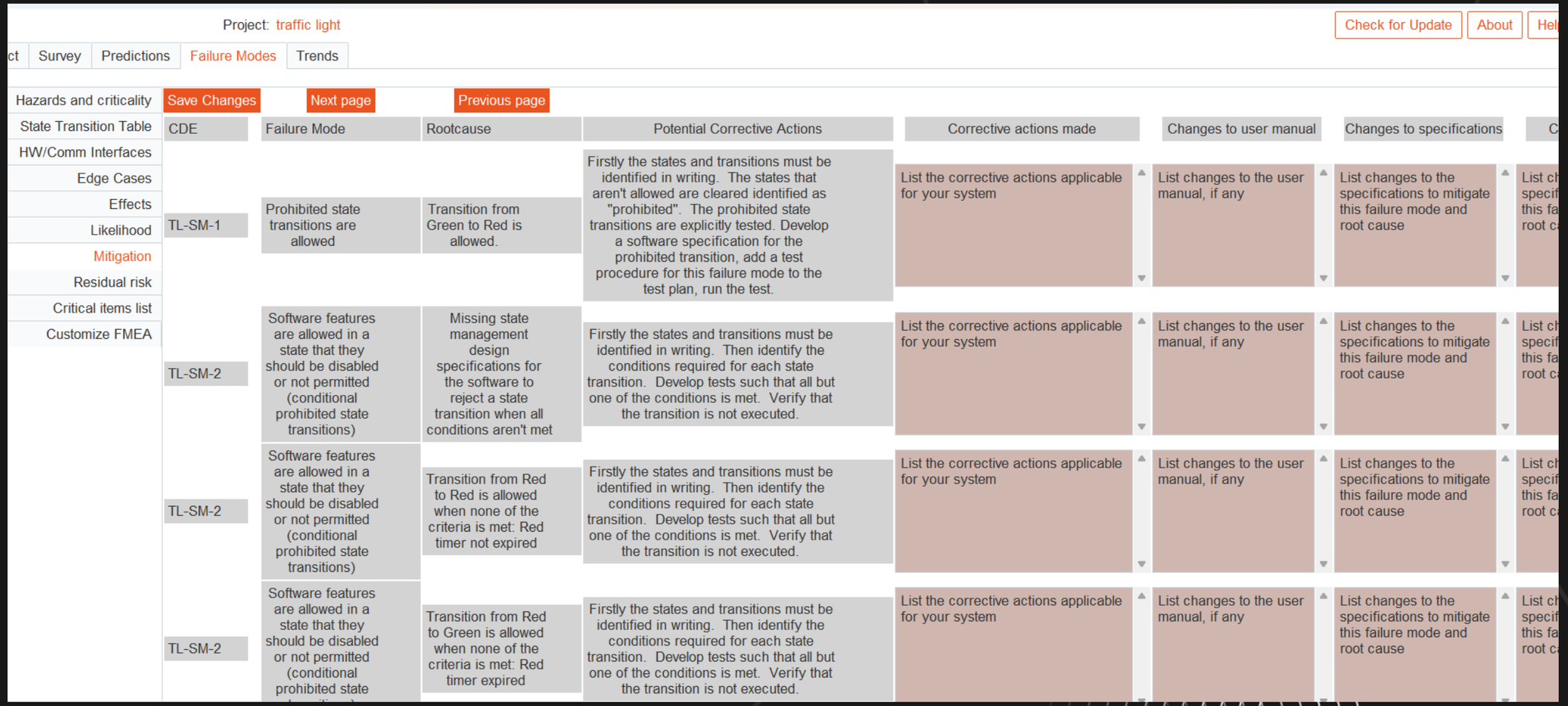
Save changes | Next page | Previous page

CDE	Failure Mode	Root Cause	Existing controls	Existence	Manifestation	Control	Detect	Average
TL-SM-1	Prohibited state transitions are allowed	Transition from Green to Red is allowed.	List preventive measures that are currently in place. The corrective action column is for preventive measures that aren't yet in place. Don't list compensating provisions by the end user in this column.	1.75	2.5	2.5	1.0	1.94
TL-SM-2	Software features are allowed in a state that they should be disabled or not permitted (conditional prohibited state transitions)	Missing state management design specifications for the software to reject a state transition when all conditions aren't met	List preventive measures that are currently in place. The corrective action column is for preventive measures that aren't yet in place. Don't list compensating provisions by the end user in this column.	1.75	2.5	2.5	1.0	1.94
TL-SM-2	Software features are allowed in a state that they should be disabled or not permitted (conditional prohibited state transitions)	Transition from Red to Red is allowed when none of the criteria is met: Red timer not expired	List preventive measures that are currently in place. The corrective action column is for preventive measures that aren't yet in place. Don't list compensating provisions by the end user in this column.	1.75	2.5	2.5	1.0	1.94
TL-SM-2	Software features are allowed in a state that they should be disabled or not permitted (conditional prohibited state	Transition from Red to Green is allowed when none of the criteria is met: Red timer expired	List preventive measures that are currently in place. The corrective action column is for preventive measures that aren't yet in place. Don't list compensating provisions by the end user in this column.	1.75	2.5	2.5	1.0	1.94

Hardware methods don’t work for software. You will learn the industry standard method for Software..

Test and mitigation strategies

The entire purpose of the software FMEA is to affect design and test cases. We show you how to do that.



Assessment and Mitigation Strategies

Assess Likelihood with Objective Measures

Hardware methods don't work for software failures. Learn the industry standard method for relative likelihood assessment.

- Industry standard methodology specifically designed for software
- Objective measures replace subjective hardware-based approaches
- Relative likelihood assessment provides meaningful risk prioritization

Test and Mitigation Implementation

The entire purpose of software FMEA is to affect design and test cases effectively for comprehensive coverage.

- Direct impact on design decisions and architectural choices
- Improved test case development for comprehensive coverage
- Practical mitigation strategies with closure tracking systems

Who Needs This Training

Reliability Engineers

- Learn fastest way to complete software FMEAs
- Avoid falling behind schedule in Agile environment
- Maintain project timelines with efficient methodology

Safety Engineers

- Use software FMEA as starting point for hazards analysis
- Cover both safety and availability failures efficiently
- Integrate with existing safety management systems

Software Engineers & Testers

- Find design issues early in development cycle
- Use CDEs to improve design and reduce technical debt
- Develop better test procedures for fault injection

Ready to Get Started?

Contact us today to schedule your Software FMEA training: sales@missionreadysoftware.com

Mission Ready Software