Workshop TSF Linking Evidences: Driving Automotive Software Safety

DrivaPi Team04

05-11-2025

Empowering the DrivaPi team to master TSF concepts, ISO 26262 standards, and practical workflows for effective software safety verification

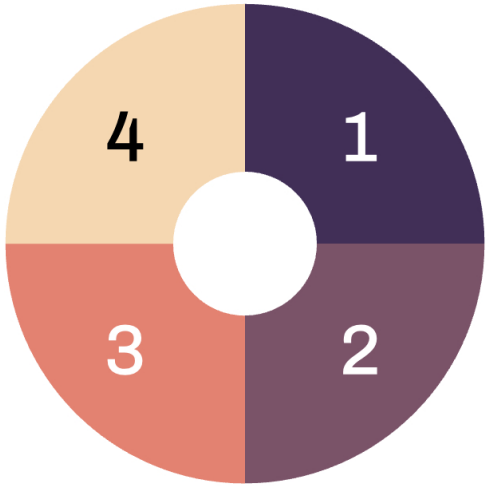
sea|me
software engineering in automotive and mobility ecosystems

Core TSF Concepts Review

Understanding Traceability, Evidence, and Confidence in Software Safety

Methodology Steps

Defines expectations, collects evidence, documents assumptions, builds arguments, and evaluates confidence for software safety assurance.



Statements

Concise assertions representing requirements or claims forming the basis of traceability.

Evidence

Concrete artifacts such as code, tests, and documents that support or verify statements.

Links

Relationships connecting statements to each other and to evidence, enabling traceability.

TSF: Key Evidence Domains

Exploring Six Core Areas Driving Software Reliability

Origin & Integrity

Provenance: Software origin and authorship

Construction: Build and installation integrity

Function & Performance

Expectations: Intended functions and constraints

Results: Actual test outcomes and metrics

Maintenance & Trust

Change: Updates and regression management

Confidence: Overall trust from domain evaluations

ASIL Levels Explained

Understanding Risk Classification and DrivaPi Safety Context

ASIL Calculation Framework

Assessing Severity, Exposure, and Controllability to Define Risk Levels

Severity (S) Measures Potential Harm Level, Rated From S0 To S3

Exposure (E) Reflects How Frequently Risk Conditions Occur, Ranging E0 To E4

Controllability (C) Assesses Ability To Prevent Harm, Rated C1 To C3

ASIL Is Calculated By Combining S, E, And C In A Risk Matrix, Producing Levels QM, A, B, C, Or D

Example: Emergency Brake With Fatal Severity (S3), Frequent Exposure (E4), And Low Controllability (C3) Yields Highest ASIL D

Severity (S) Factor Defines Injury Impact

Understanding severity levels clarifies risk and guides ASIL classification in DrivaPi

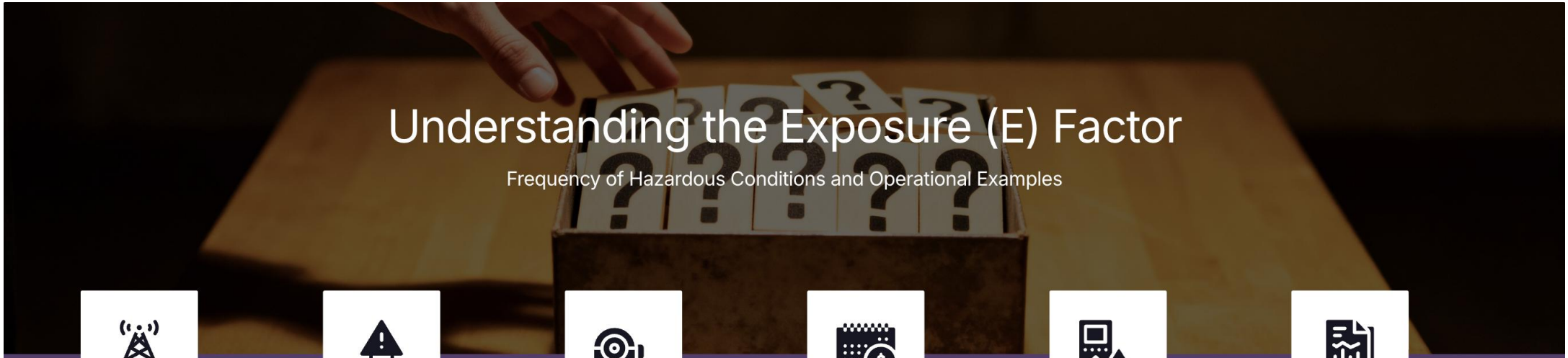
Severity directly influences ASIL rating; only **S3** is used for critical production controls

S3: Fatal or life-threatening injuries; critical production system functions like emergency stop

S0: No injury possible; negligible risk, such as simple display widgets

S1: Minor to moderate injuries (scratches, sprains); low severity use cases

S2: Severe but survivable injuries (fractures, concussions); medium risk scenarios



Understanding the Exposure (E) Factor

Frequency of Hazardous Conditions and Operational Examples

E0: Extremely unlikely – hazard occurs less than 0.001% of operating time

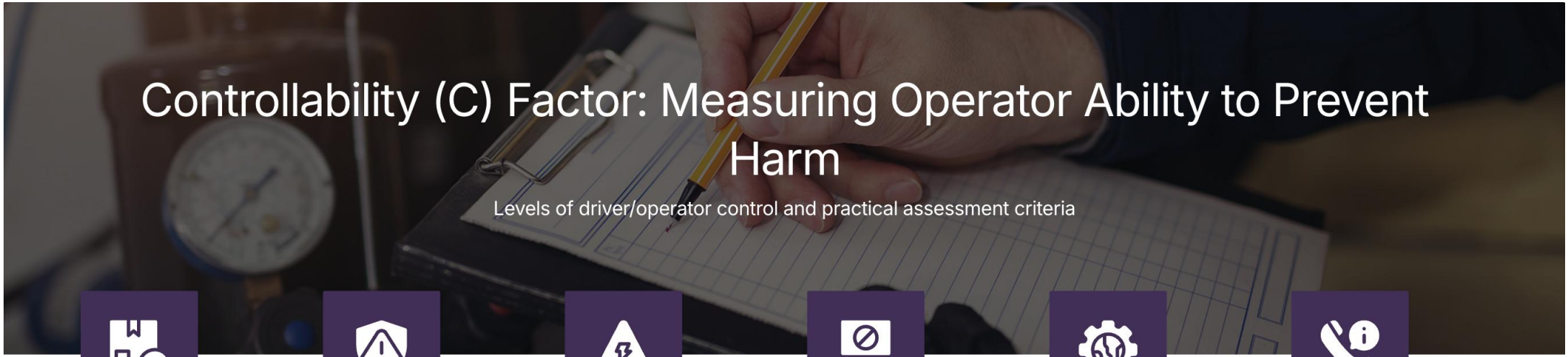
E1: Very low probability – hazard arises less than 1% of operating time

E2: Low probability – hazard appears between 1% and 10% of operating time

E3: Medium probability – about 10% operating time, e.g., hazard likely once per month

E4: High probability – hazard occurs more than 10% of operating time

Exposure is assessed via qualitative and quantitative methods using operational data and analogies to estimate hazard frequency.



Controllability (C) Factor: Measuring Operator Ability to Prevent Harm

Levels of driver/operator control and practical assessment criteria

C0: Easy control – Immediate operator action possible without difficulty

C1: Moderate control – Reasonable measures needed to maintain control, e.g., early warnings enabling safe braking

C2: Difficult control – Significant challenges exist, requiring complex operator efforts

C3: Very difficult control – Little or no control possible by the operator

Assessment factors include warning signals, operator workload, lighting, and environmental conditions to gauge control level accurately

Example: Early collision warnings enabling safe braking classify as **C1** controllability level

Severity	Exposure	Controllability		
		C1 (Simple)	**C2** (Normal)	**C3** (Difficult, Uncontrollable)
S1 LIGHT AND MODERATE INJURIES	**E1** (Very low)	QM	QM	QM
	E2 (Low)	QM	QM	QM
	E3 (Medium)	QM	QM	A
	E4 (High)	QM	A	B
S2 SEVERE AND LIFE THREATENING INJURIES – SURVIVAL PROBABLE	**E1** (Very low)	QM	QM	QM
	E2 (Low)	QM	QM	A
	E3 (Medium)	QM	A	B
	E4 (High)	A	B	C
S3 LIFE THREATENING INJURIES, FATAL INJURIES	**E1** (Very low)	QM	QM	A
	E2 (Low)	QM	A	B
	E3 (Medium)	A	B	C
	E4 (High)	B	C	D

QM (Quality Management)
Development supported by established Quality Management is sufficient.

A
B
C
D

lowest ASIL
Low risk reduction necessary

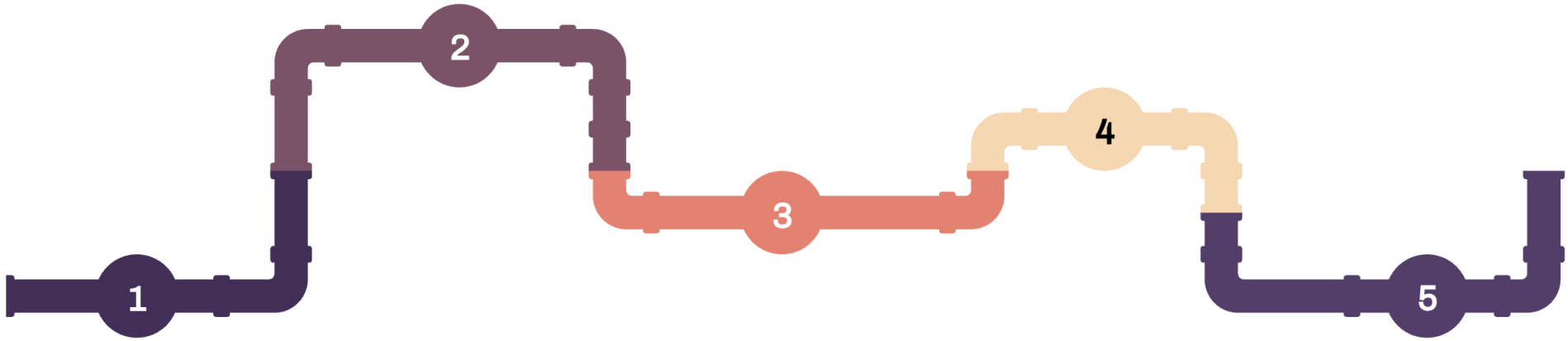
highest ASIL
High risk reduction necessary

sea|me
software engineering in automotive and mobility ecosystems

Functional Safety Hazard Analysis and Risk AssessmentHARA Documentation Model

HARA Documentation Model

Systematically identify hazards and assign safety goals with a clear example from DrivaPi



Define System Scope

Determine system boundaries to focus hazard analysis on relevant components.

Identify Potential Hazards

Use checklists to list hazards that may impact system safety.

Assess Risks with ASIL Matrix

Evaluate hazards by Severity, Exposure, and Controllability to assign ASIL levels.

Set Safety Goals

Define safety goals that mitigate risks, ensuring traceability and justification.

DrivaPi Motor Speed Sensor Example

Detects reading errors as hazards with moderate risk; safety goal requires 1-second alert.

Evidence Linking with Only References Block

Reliable syntax aligned with ISO 26262 manual review

```yaml
header: "Motor Speed Sensor Driver Implementation"
text: |
  "Implement GPIO-based motor speed sensor driver that reads pulse count
  over 1-second intervals and converts to RPM with range validation."

# TSF Type: Assertion (Both a Request and a Claim)

ASIL: "A"
verification_method: "Unit Testing, Code Review, Static Analysis"

# Links: Connects to parent Assertion
parents:
  - id: SRD-010

# Links: Connects to child Evidence
children:
  - id: LLTC-010

reviewers:
  - name: "Carol Dev"
    email: "carol@team.com"

reviewed: ''  # Manually fill on PR approval (YYYY-MM-DD - Approved by Name <email>)

# Evidence Linking (use 'references:')
references:
  - type: "file"
    path: "src/sensors/motor_speed.cpp"
    description: "Driver implementation"
  - type: "file"
    path: "src/sensors/motor_speed.h"
    description: "Driver header file"
  - type: "file"
    path: "artifacts/verification/static-analysis/cppcheck-SWD-010.xml"
    description: "Static analysis results: 0 errors, 0 warnings"
  - type: "file"
    path: "artifacts/verification/coverage/coverage-SWD-010.xml"
    description: "Code coverage: 87%"
  - type: "file"
    path: "docs/standards/iso26262/hara-motor-speed.md"
    description: "HARA Analysis"
  - type: "file"
    path: "docs/standards/iso26262/asil-justification-SWD-998.md"
    description: "ASIL A determination"

active: true
derived: false
normative: true
level: 3.0  # 1.0=URD, 2.0=SRD, 3.0=SWD, 4.0=LLTC
---
```

Evidence linking uses only the **references:** block, excluding **evidence:** block to avoid complexity (for now)

Example of references block listing files from source code, unit tests, and ISO 26262 HARA documentation

Manual SME review is mandatory per ISO 26262 to ensure expert judgment without automation risks

Ensures compliance with ISO 26262 ASIL A+ requirements

Score Location: ONLY at LLTC Level

Assign score exclusively in LLTC to ensure clear trust propagation across all documents

Score Placement as an Umbrella

Placing scores only at the LLTC level acts like an umbrella covering the entire feature scope, shielding upstream documents (SWD, SRD, URD) that inherit a consistent score automatically.

Clear Trust Propagation Path

By concentrating scoring in LLTC, the reviewer assesses the complete chain once. This ensures consistent trust evaluation and avoids conflicting or multiple scores across documents.

sea|me
software engineering in automotive and mobility ecosystems

ASIL Field Placement: Only in SWD for Safety Clarity

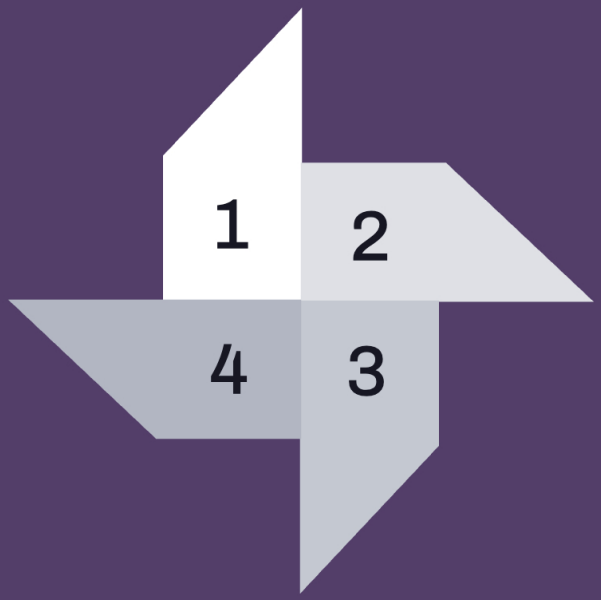
Ensuring ISO 26262 Compliance by Centralizing ASIL in Software Design

User Requirements Document (URD)

No ASIL field here. Focuses on user needs, not implementation safety. Serves as the feature's functional foundation.

Low-Level Test Cases (LLTC)

No ASIL field. Inherits ASIL from SWD parent document, ensuring tests align with safety-critical implementation.



System Requirements Document (SRD)

Also excludes ASIL field. Defines system needs without implementation details. Builds on URD for system functionality.

Software Design Document (SWD)

Contains the ONLY ASIL field, e.g., ASIL: "A". Critical for software implementation safety compliance under ISO 26262.

Lab 1: Creating and Reviewing Requirements

Key Tasks and Workflow for Effective Requirement Development in TSF

- Understand and differentiate URD-999 (unit requirements) and SRD-999 (system requirements)
- Define monitoring requirements focused on battery components
- Establish clear preconditions and postconditions for requirements
- Link URD and SRD through traceability to ensure consistency
- Prepare test cases integrating safety considerations
- Collaborate in pairs using git version control for traceability and teamwork
- Estimated lab time is 45 minutes, focused on hands-on practice

sea|me
software engineering in automotive and mobility ecosystems

Lab 2: ASIL Assessment and Evidence Linking

Complete SWD-999 with Evidence and ASIL to Ensure Compliance

Create HARA Document

Identify hazards and calculate ASIL using S×E×C factors. (30 min)

Write ASIL Justification

Explain ASIL level A and compile supporting evidence. (30 min)

Complete SWD-999

Add 10 references, input ASIL field, and validate with trudag lint tool. (30 min)

Add Score in URD-999

Reviewer assigns confidence score; verify correct score propagation. (20 min)

Generate Report

Publish final report and review dashboard status for completeness. (10 min)

Seamless CI/CD Pipeline Integration

Automate TSF Validation and Continuous Compliance with GitHub Actions

Run Automated Tests

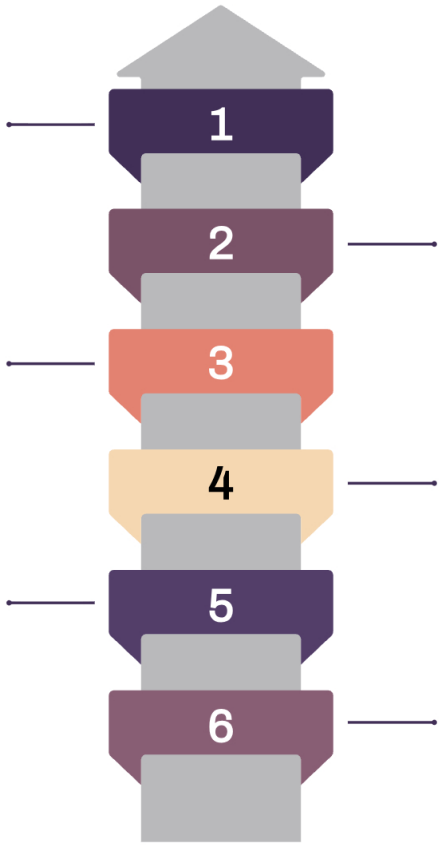
Execute tests generating LLTC-999-junit.xml with pass/fail results.

Measure Coverage

Generate coverage reports such as motor-coverage.txt to assess test completeness.

Publish Reports

Publish trustworthy reports to the reviewer dashboard for continuous compliance visibility.



Perform Static Analysis

Run cppcheck to produce detailed static analysis reports.

Validate and Score

Run trudag lint for structure validation and trudag score to assess confidence levels.

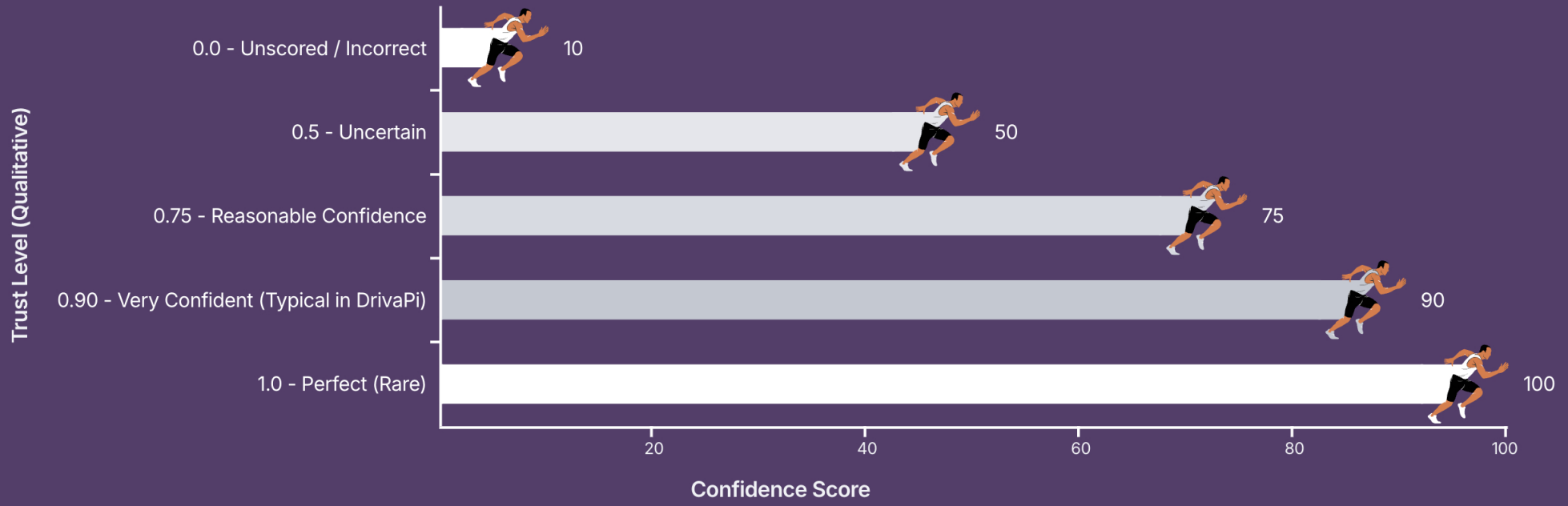
Commit Artifacts

Store validation artifacts in artifacts/verification/ and finalized reports in artifacts/trustable-report/.

sea|me
software engineering in automotive and mobility ecosystems

Interpreting Confidence Scores for Safety Assurance

Understand trust levels with a clear, stepwise confidence scale



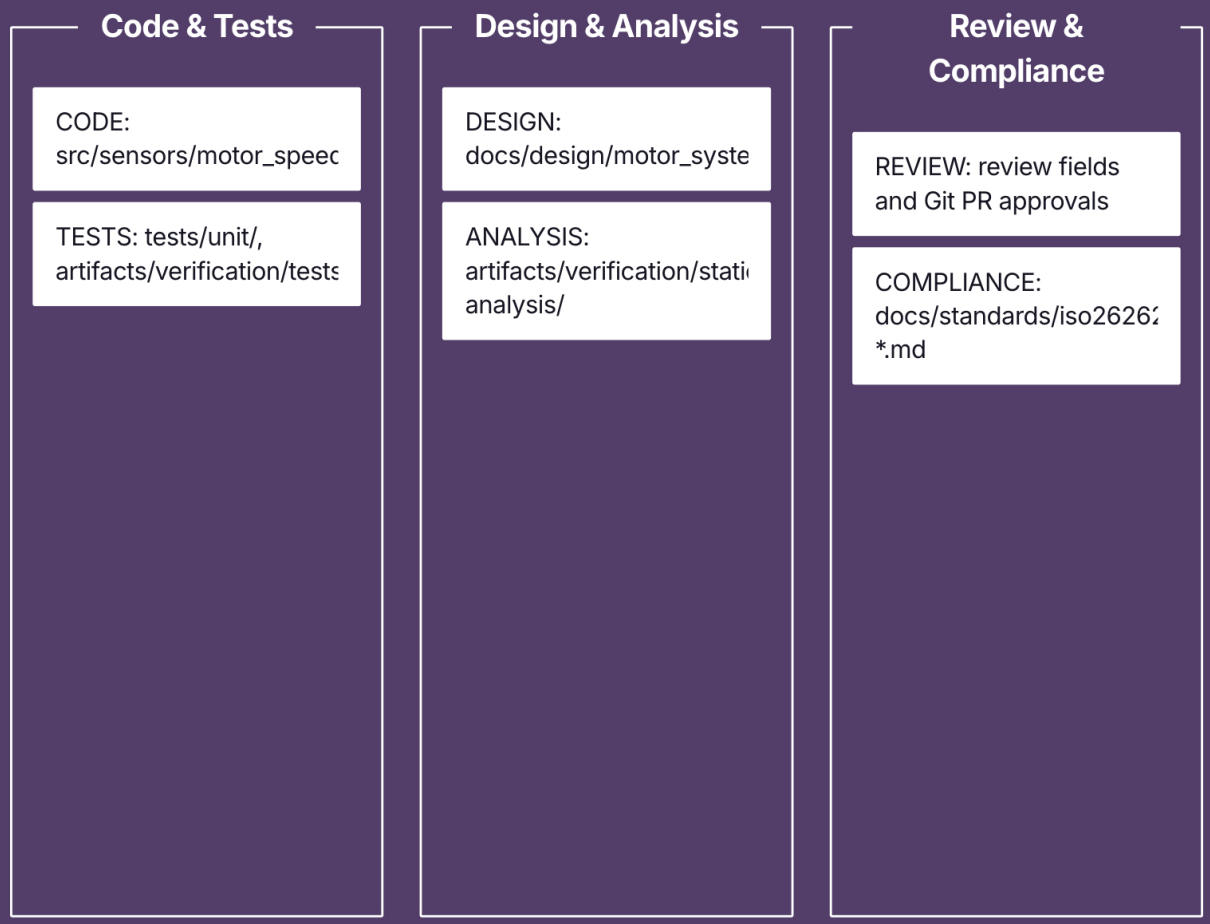
DrivaPi ASIL Assignment by Feature

Aligning Safety Levels to Feature Criticality for Focused Development

Feature	ASIL	Rationale
Display widgets	QM	Non-safety function
Temperature sensor	A	Low risk monitoring
Motor speed sensor	B	Medium risk
Emergency stop	C	Safety-critical function

Mapping Evidence Types to DrivaPi Repositories

Visualize where key evidence domains reside to enhance traceability and audit readiness



Q&A and Discussion

Explore key topics and clarify evidence linking and ASIL practices

Correct evidence linking practices in TSF

Challenges faced in DrivaPi implementation

Best practices for HARA documentation

Determining correct ASIL levels

Maintaining traceability in complex projects

Integrating CI/CD with TSF workflows

Evolving ASIL practices and linkage

Active participation ensures knowledge consolidation and practical understanding

sea|me
software engineering in automotive and mobility ecosystems

Next Steps to TSF Reviewer Certification

Sequential actions post-workshop to ensure compliance and certification success

Apply Linking Methods

Implement linking syntax and references within the DrivaPi repository for traceability.

Complete Lab Exercises

Finish lab tasks including evaluations to reinforce workshop concepts and demonstrate proficiency.

Implement ASIL Framework

Integrate ASIL classification within software components for ISO 26262 compliance.

Create TSF Documentation Models

Develop thorough TSF documentation to support evidence domains and audit requirements.

Achieve TSF Reviewer "Certification"

Obtain "certification" after successful project approval and meeting all compliance criteria.

Integrate CI/CD Pipeline

Leverage templates and validation tools in CI/CD for continuous compliance and quality assurance.

References and Credits

Acknowledging Contributions and Resources

sea:me

software engineering in automotive and mobility ecosystems

Thank you for participating in the DrivaPi TSF Evidence Linking and ASIL Justification Workshop

For detailed information and materials, refer to the project repository documentation

Presented by the DrivaPi Team

Link to original materials available upon request