

Trustable Software Framework (TSF)

Fundamentals and Practical Application for Automotive Software

A comprehensive approach to software trustworthiness



Graph Model



Requirements



Traceability

Learning Goals

By the end of this training, participants will be able to:



Understand TSF Fundamentals

Grasp core principles of the Trustable Software Framework and its application to automotive software development



Awareness of Standards

Develop understanding of ISO 26262 and ASPICE as they relate to verification efforts and safety-critical systems



Write Traceable Requirements

Learn to create clear, testable requirements with proper traceability for critical automotive systems



Key Focus: Practical application of TSF principles to automotive software development, including repository layout, workflow implementation, and evidence collection.

Why TSF Matters

"TSF is crucial for managing the increasing complexity of modern software by making claims and requirements explicit and linking them directly to verifiable evidence."



Explicit Requirements



Makes requirements and claims explicit, linking them directly to verifiable evidence

Confidence Assessment



Allows assessors to objectively evaluate trustworthiness through evidence-based confidence scoring

Risk Management



Enables measurement, management, and reduction of risks in complex software releases

Safety-Critical Focus



Essential for safety, security, performance, availability, and reliability in automotive systems

TSF Model and Methodology

Model Components



Statements

Concise, affirmative claims or requirements that form the nodes of the graph, written to support logical reasoning



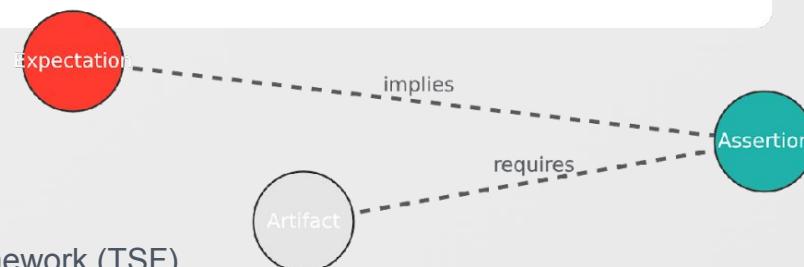
Links

Define implication relationships between Statements, indicating how one Statement supports or is derived from another



Artifacts

Concrete pieces of evidence such as source code, test results, and documentation that support and validate Statements



TSF Methodology

Setting Expectations

Defining critical functional and non-functional requirements as Expectations (requests without parents)

Providing Evidence

Gathering artifacts to support claims, creating links between Statements and evidence

Documenting Assumptions

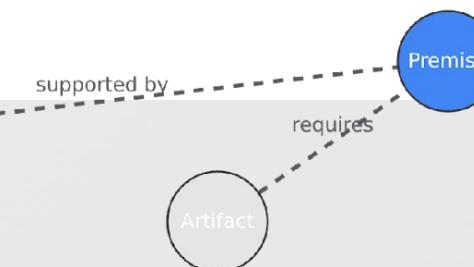
Recording external dependencies as "dangling" Premises (claims without children)

Recording Reasoning

Building logical arguments using Assertions (both requests and claims)

Assessing Confidence

Scoring evidence and recursively calculating confidence for all Statements



Key Evidence Areas

TSF requires evidence collection across these six key areas to establish trustworthiness



Provenance

Origin of software, claims about source and licensing, understanding who produced it



Construction

How software is built and installed, ensuring correctness in these processes



Change

Management of updates and verification to prevent regressions or side effects



Expectations

What software is designed to do and what it must not do, covering functional and non-functional requirements



Results

Actual behavior compared to expectations through testing and performance metrics



Confidence

Overall assessment of software trustworthiness, derived from evaluation of all evidence areas

Requirements and Repository Layout

Repository Structure

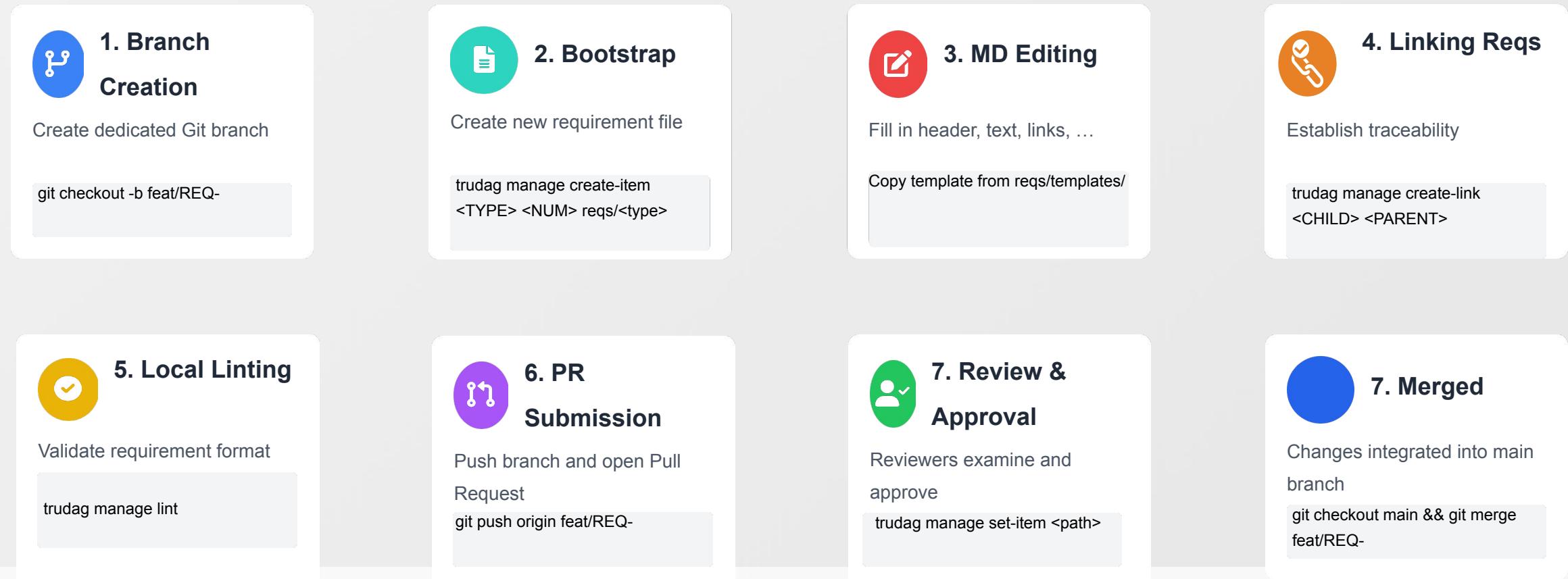
 reqs/ Dedicated directory for all requirements
 urd/ User Requirements (high-level user needs)
 srd/ System Requirements (system functionalities)
 swd/ Software Requirements (software behavior)
 lltc/ Low-Level Test Cases (test verification)

MD Template Fields

 ref Canonical reference matching file ID
 header Concise, one-line title for the requirement
 text Detailed, testable description of the requirement
 reviewers List of individuals responsible for review
 reviewed Records provenance of approval (Git SHA)
 links References to related requirements or artifacts

Authoring and Review Workflow

A systematic approach to requirement authoring with quality and traceability



i Key Point: The "reviewed:" field in the MD file records approval provenance as a Git SHA, enabling clear audit trails.

Authoring and Review Workflow - continuation

A systematic approach to requirement authoring with quality and traceability



9. Calculate Score

Determine score trustability

```
trudag score
```



11. Report

Generate report and artifacts
that support the evidence

```
trudag publish --output-dir  
artifacts/trustable-report
```



12. Baseline

Snapshot of a current state

```
git tag -a BASELINE-V1.0 -m  
"Sprint 1 baseline"
```

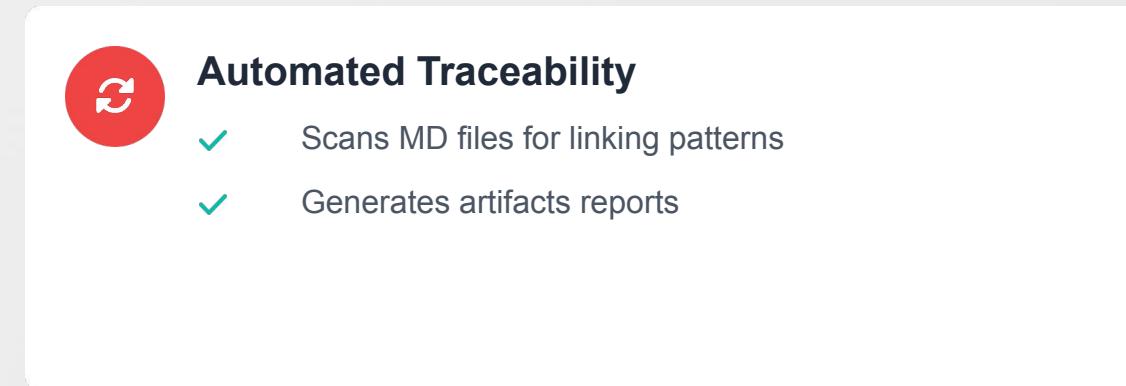
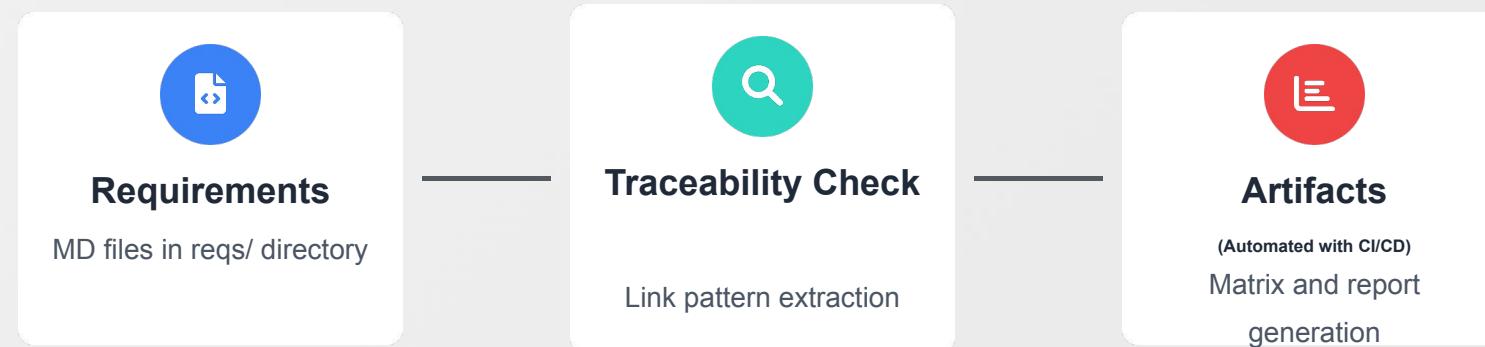


13. Keep record

Save Baseline in folder

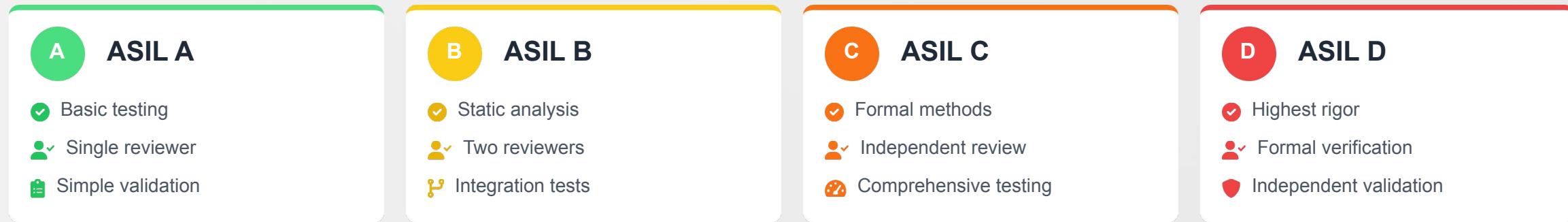
```
mkdir -p artifacts/baselines  
cp -r artifacts/trustable-report  
artifacts/baselines/v1.0-$(date  
+%Y%m%d)
```

Traceability and Continuous Integration



V&V and ASIL Mapping

Verification and Validation activities scale with ASIL levels to ensure appropriate safety assurance



Key V&V Activities by ASIL Level

ASIL B/C/D: Static analysis mandated to identify potential code defects

High-Risk Claims: Formal methods considered for mathematical proof of correctness



Hands-on Lab and Assessment



Lab Exercises



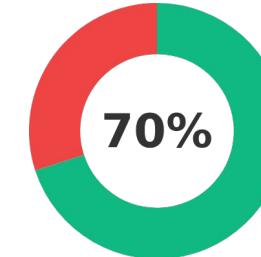
- 📄 Author a new requirement with proper ref, header, and text
- 🔗 Link to upstream design and downstream test case
- </> Run linter to check formatting and structure
- 🔍 Execute traceability checker and inspect artifacts report

2 session



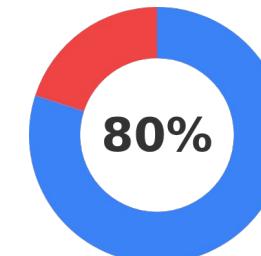
Assessment

Knowledge Quiz



- ✓ 10 questions
- ✓ 70% passing threshold
- ✓ 90% indicates high confidence

Lab Evaluation



- ≡ 5 tasks total
- ≡ 4 tasks required for passing
- 🌟 Completion certifies you as a TSF reviewer