TSF Implementation Guide Package

Project: DrivaPi Team04 - TSF Completion
Date: October 17, 2025
Package Version: 1.0

Package Overview

This implementation package contains everything you need to complete the remaining 15% of your TSF implementation and achieve 100% compliance.

What's Included:

1. **TSF Team Assessment Quiz & Lab** - Validate team understanding
2. **TSF Quick Reference Card** - Daily use reference
3. **Artifact Linking Guide** - Step-by-step linking instructions
4. **Review & Approval Workflow** - Standardized review process
5. **3-Week Completion Plan** - Day-by-day implementation schedule

Current Status Summary

Overall Completion: 85%

User Story	Status	Completion	Critical Gaps
#17 TSF Training	✅ Excellent	90%	Team assessment quiz
#18 Requirements	✅ Strong	85%	SME approval needed
#19 Traceability	✅ Excellent	90%	Artifact links

What You've Achieved (Outstanding Work!)

✅ **Infrastructure Excellence**

- Complete requirements management structure (`reqs/` with URD/SRD/SWD/LLTC)
- Doorstop + trudag tooling configured with Docker
- CI/CD pipeline automating validation
- Comprehensive training materials

✅ **Documentation Quality**

- Professional training materials covering TSF, ISO 26262, ASPICE
- Templates with mandatory fields and explanations

- Review checklists and process documentation
- V&V plan mapped to ASIL levels

✓ **Traceability Framework**

- Automated bidirectional traceability matrices
- Forward and backward link generation
- Gap reporting identifying missing links
- No broken links or orphaned requirements

What Remains: The Final 15%

High Priority (Must Complete)

1. Team Assessment (Story #17)

Gap: No formal assessment recorded
Solution: Use `TSF-Team-Assessment.md`
Time: 4 hours (1 hour per team member)
Impact: Validates team competency, completes DoD

Action Plan:

- Schedule assessment session Monday morning
- Each team member takes 30-min quiz + 30-min lab
- Grade and record results
- Update Definition of Done: "Assessment confirms understanding" ✓

2. Requirement Approval (Story #18)

Gap: Requirements show `reviewed: ''` (not approved)
Solution: Use `Review-Approval-Workflow.md`
Time: 4 hours for 8 requirements
Impact: Completes baseline approval process

Current Requirements Needing Approval:

- URD-001 (User requirement)
- SRD-001 (System requirement)
- SWD-001, SWD-002 (Software requirements)
- LLTC-001 (Test case)

Action Plan:

```
# For each requirement after review:
git log -1 --format=%H  # Get SHA
# Edit YAML: reviewed: &lt;sha&gt;
git commit -m "review: Approve [REQ-ID]"
```

3. Artifact Linking (Story #19)

Gap: All 8 requirements missing artifact links
Solution: Use `Artifact-Linking-Guide.md`
Time: 2 weeks (phased approach)
Impact: Achieves full traceability, closes gaps

Artifacts to Create/Link:

- Design documents (architecture, component design)
- Source code files (existing code)
- Test specifications and implementations

Action Plan:

- Week 1: Create architecture docs, link to SRD
- Week 2: Link code to SWD, tests to LLTC
- Validate: `python scripts/verify_artifacts.py`

Implementation Timeline (3 Weeks)

Week 1: Foundation (Oct 21-25)

Focus: Assessment, Approval, Architecture

Monday: Team assessment + grading
Tuesday: Approve all existing requirements
Wednesday: Create architecture documents
Thursday: Link SRD to architecture
Friday: Link SWD to code, LLTC to tests

Deliverables:

- ✅ Team assessed and competent
- ✅ All requirements approved
- ✅ Architecture documented and linked

Week 2: Expansion (Oct 28 - Nov 1)

Focus: New Requirements, Expanded Coverage

Monday: Update templates with artifact sections
Tuesday: Create 5 new URD requirements
Wednesday: Create corresponding SRD requirements
Thursday: Create SWD requirements with code links
Friday: Create LLTC test cases

Deliverables:

- ✅ 20+ new requirements created
- ✅ Full URD→SRD→SWD→LLTC chains
- ✅ Templates updated for future use

Week 3: Validation & Baseline (Nov 4-7)

Focus: Validation, Baseline, Completion

Monday: Create missing artifacts (stubs if needed)
Tuesday: Run full validation suite
Wednesday: Create Baseline v2.0
Thursday: Final documentation + retrospective

Deliverables:

- ✅ Zero validation errors
- ✅ Zero gaps in gap-report.csv
- ✅ Baseline v2.0 created and tagged
- ✅ 100% TSF compliance achieved

Quick Start: What to Do Monday Morning

Step 1: Distribute Materials (30 minutes)

```
# Save all documents to your repo
mkdir -p docs/implementation/
cp TSF-Team-Assessment.md docs/implementation/
cp TSF-Quick-Reference.md docs/implementation/
cp Artifact-Linking-Guide.md docs/implementation/
cp Review-Approval-Workflow.md docs/implementation/
cp TSF-Completion-Plan.md docs/implementation/

git add docs/implementation/
```

```
git commit -m "docs: Add TSF implementation completion package"
git push
```

Step 2: Team Standup (15 minutes)

- Review completion plan
- Assign roles (who does what)
- Schedule team assessment for Monday 10:00 AM

Step 3: Conduct Team Assessment (4 hours)

- Use `TSF-Team-Assessment.md`
- Each member: 30 min quiz + 30 min lab
- Grade and record results
- Update team competency tracking

Step 4: Approve Requirements (Afternoon)

- Follow `Review-Approval-Workflow.md`
- Review existing requirements
- Set `reviewed:` field with git SHA
- Commit approvals

Success Metrics

By November 7, you will have:

✓ **100% Team Competency**

- All members assessed (≥80% passing)
- Quick reference cards distributed
- TSF principles demonstrated

✓ **Complete Requirements Coverage**

- 25+ requirements defined (up from 8)
- All requirements approved with git SHA
- Baseline v2.0 created and tagged

✓ **Full Traceability**

- 100% requirement coverage
- All artifacts linked (design, code, tests)
- Zero gaps in gap-report.csv

- Bidirectional traceability functional

✅ **TSF Compliance Achieved**

- All Definition of Done items complete
- All Acceptance Criteria met
- Ready for automotive-grade development
- Team demonstrates TSF competency

Daily Commands Reference

Every Morning

```
# Check status
python scripts/reqs_lint.py
doorstop
python scripts/traceability_check.py
cat artifacts/traceability/gap-report.csv
```

Creating Requirements

```
# Copy template
cp reqs/templates/URD-000.yml reqs/urd/URD-XXX.yml

# Edit and validate
nano reqs/urd/URD-XXX.yml
python scripts/reqs_lint.py

# Commit
git add reqs/urd/URD-XXX.yml
git commit -m "feat: Add URD-XXX - [description]"
```

Approving Requirements

```
# Get SHA
SHA=$(git log -1 --format=%H)

# Update reviewed field, then:
git add reqs/[category]/[REQ-ID].yml
git commit -m "review: Approve [REQ-ID]"
```

Checking Progress

```
# Count requirements
find reqs/ -name "*.yml" -not -name "*-000.yml" | wc -l

# Check approved
grep -r "reviewed: ''" reqs/*/

# Check gaps
cat artifacts/traceability/gap-report.csv
```

Risk Mitigation

If team members fail assessment:

- Allow open-book quiz
- Provide immediate remedial training
- Allow retake after 2 days
- Focus on understanding, not memorization

If artifacts are missing:

- Create stub files as placeholders
- Document TODOs clearly
- Link stubs to requirements
- Plan implementation incrementally

If validation fails:

- Run validation daily
- Fix issues immediately
- Keep scripts updated
- Don't accumulate technical debt

If timeline slips:

- Prioritize critical requirements first
- Use templates for consistency
- Parallelize work where possible
- Request help from team lead

Support Resources

Documentation

- **Training:** `docs/tsf/TSF-training.md`
- **Templates:** `docs/tsf/TSF-requirement-template.md`
- **Review Process:** `docs/requirements/REVIEWER_CHECKLIST.md`
- **V&V Plan:** `docs/tsf/VV-plan.md`

Scripts

- **Linter:** `python scripts/reqs_lint.py`
- **Traceability:** `python scripts/traceability_check.py`
- **Doorstop:** `doorstop`

Implementation Guides (This Package)

- **Assessment:** `TSF-Team-Assessment.md`
- **Quick Reference:** `TSF-Quick-Reference.md`
- **Artifact Linking:** `Artifact-Linking-Guide.md`
- **Review Workflow:** `Review-Approval-Workflow.md`
- **Completion Plan:** `TSF-Completion-Plan.md`

Quality Gates

Before Week 1 Ends:

- [ ] All team members assessed
- [ ] All existing requirements approved
- [ ] Architecture documents created

Before Week 2 Ends:

- [ ] 15+ new requirements created
- [ ] All requirements linked to parents
- [ ] Templates updated

Before Week 3 Ends:

- [ ] All artifacts created/linked
- [ ] Validation passes cleanly
- [ ] Baseline v2.0 created
- [ ] Gap report empty

Final Completion:

- [ ] 100% traceability coverage
- [ ] Zero validation errors
- [ ] All DoD items complete
- [ ] Team retrospective done

Troubleshooting

Problem: Linter errors
Solution: `python scripts/reqs_lint.py` shows specific issues

Problem: Broken parent links
Solution: Verify parent requirement exists and ref is correct

Problem: Missing artifacts
Solution: Create stub files, use `verify_artifacts.py` script

Problem: Time constraints
Solution: Focus on critical path items first, parallelize work

Problem: Team questions
Solution: Review training materials, escalate to team lead

Completion Checklist

Story #17: TSF Training

- [ ] Team assessment completed (all members ≥80%)
- [ ] Quick reference card distributed
- [ ] Training materials published
- [ ] Definition of Done: Assessment confirms understanding ✅

Story #18: Requirements Definition

- [ ] All requirements approved (reviewed field set)
- [ ] Baseline v2.0 created
- [ ] Requirements document complete
- [ ] Definition of Done: Team review completed ✅

Story #19: Traceability Matrix

- [ ] All artifacts linked (zero gaps)
- [ ] Matrix created with all items
- [ ] Bidirectional traceability functional
- [ ] Definition of Done: Documentation complete ✅

Epic: TSF Implementation

- [ ] All acceptance criteria met
- [ ] All definition of done items complete
- [ ] Team demonstrates TSF competency
- [ ] **100% COMPLETE**

Next Steps After Completion

Once you achieve 100% TSF compliance:

1. **Announce Success** - Share with stakeholders
2. **Maintain Process** - Keep requirements updated as code evolves
3. **Scale Usage** - Apply TSF to new features
4. **Continuous Improvement** - Refine processes based on lessons learned
5. **Share Knowledge** - Train new team members using your materials

Final Message

You've already done the hard work—you've built an **excellent TSF framework** with solid infrastructure, comprehensive training, and automated tooling. The remaining 15% is primarily execution: running assessments, approving requirements, and linking artifacts.

Follow this 3-week plan, use the implementation guides provided, and you'll achieve **100% TSF compliance** by November 7, 2025.

Your DrivaPi project will be ready for automotive-grade development!

Contact & Support

- **Process Questions:** Review implementation guides
- **Technical Issues:** Check training materials and scripts
- **Blockers:** Escalate to team lead immediately
- **Success Updates:** Share in daily standups

You've got this! Good luck with the final push!

Package Created By: AI Analysis System
Date: October 17, 2025
Version: 1.0