Beyond Benchmarks: A Case Study of LLM-Generated Verus Specification Failures on Asterinas Vostd

Dugang Liu[1], Ding Wang[1], Chao Jiang[1], Cheng Wen[1], Zhiwu Xu[1], Zhong Ming[1], Yuwei Liu[2], Xinyi Wan[2], Lin Huang[2]

[1] College of Computer Science and Software Engineering, Shenzhen University
[2] Ant Research, Ant Group

Introduction

Writing formal specifications and proofs is costly. LLMs show promise on self-contained benchmarks, but industrial verification adds constraints that benchmarks strip away: repository integration, cross-file dependencies, callee contracts, ghost state, and Entity–Model proof abstractions.

We study LLM-generated Verus specifications in Vostd, the verification codebase of the Asterinas OS kernel:

RQ1: How do industrial targets differ from benchmarks, and how well do generated specifications verify on them?
RQ2: How do failures distribute across stages, models, and targets?
RQ3: How reliable are verifier diagnostics and verifier-guided repair?

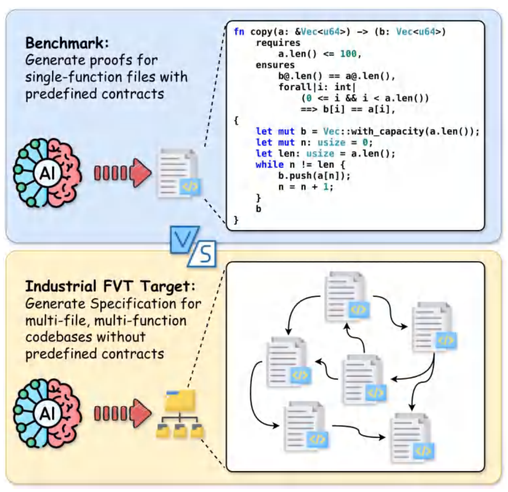


Fig. 1. Benchmark vs. industrial FVT

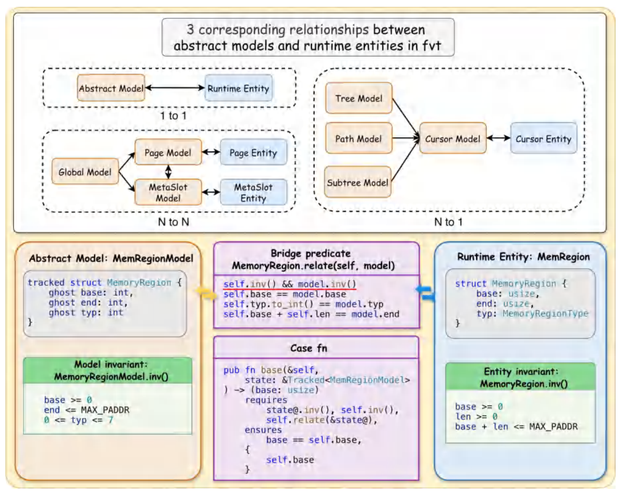


Fig. 2. Entity–Model proof style

Study Setting and Corpus

All specifications are generated, verified, and repaired inside the original Vostd repository of Asterinas OSTD, not in extracted stand-alone files.

6 FVT targets	**156** function-level tasks	**7,149** Rust LoC in 50 files
3 LLMs (Qwen, GPT-4o, DS-V4)	**270** repair traces	**1,587** reviewed failure episodes

Each failure episode gets two labels: the **visible diagnostic stage** (what Verus reports) and the **validated source stage** (where the defect really is).

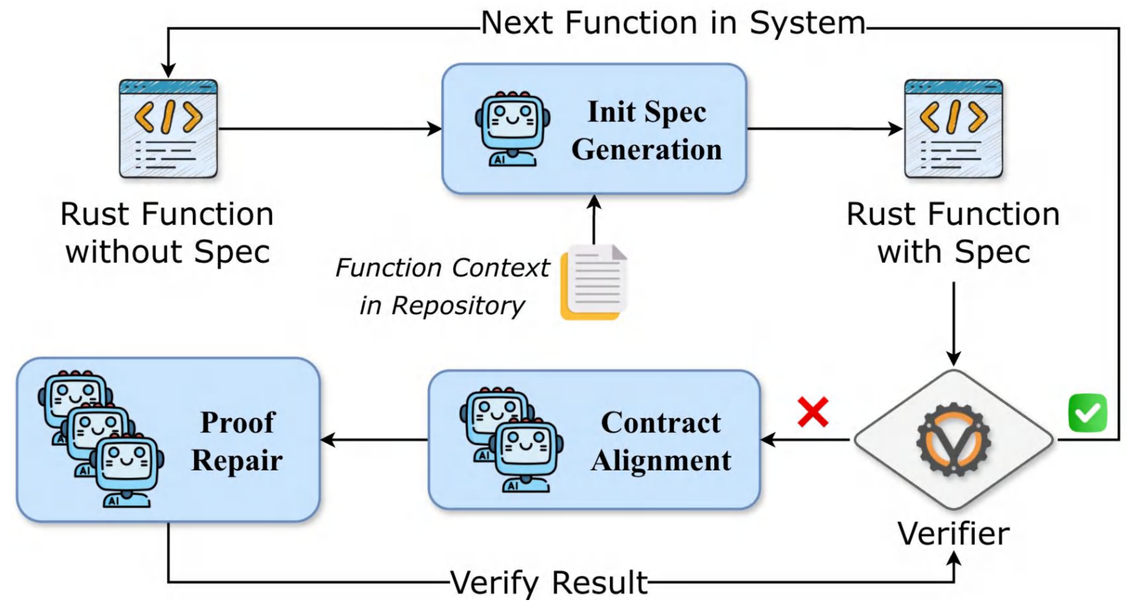


Fig. 3. Specification generation and repair

Conclusion

Evaluation: report where specifications fail (stage, diagnostic alignment, repair convergence), not only pass rates.

Tool design: attribute the source stage before repairing, and match edits to that stage.

Practice: let LLMs draft specifications with engineers in the loop; escalate repairs that stall or regress to human review.

Artifact: https://doi.org/10.5281/zenodo.21375096

Finding 1: The Benchmark-to-Deployment Gap

Benchmark success does not transfer. VeruSAGE reaches 66.0%–82.6% on benchmark tasks, but on industrial FVT, AutoVerus discharges only **22.7%–24.5%** of proof obligations and VeruSAGE **34.4%–41.7%**.

Our repository-grounded workflow, with contract alignment and proof repair, discharges **66.3%–68.7%**, back in the benchmark range.

Difficulty is driven more by the **target** (contracts, abstraction boundaries, cross-file dependencies) than by the LLM.

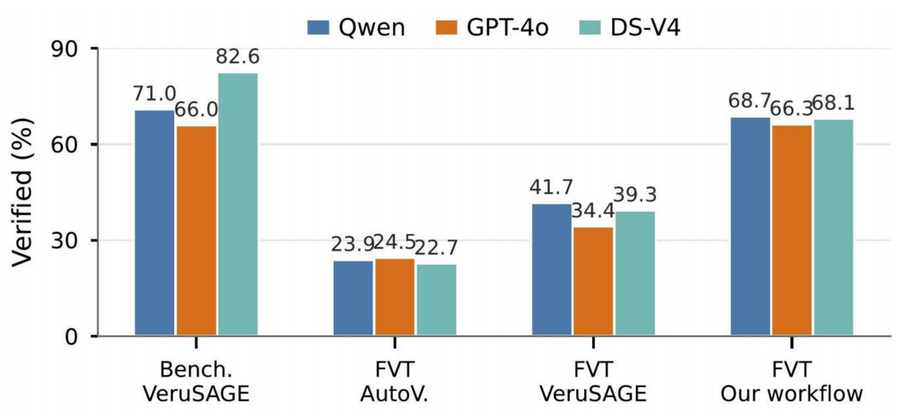


Fig. 4. Verification outcomes (benchmarks: per task; FVT: per proof obligation)

Finding 2: Failures Are Stage-Dependent

Reviewed failure episodes fall into four verification stages:

- **I. Repository integration (28%):** syntax, import, name-resolution, and hallucinated API errors.
- **II. Verus legality (32%):** valid Rust that breaks Verus mode or type rules.
- **III. Semantic modeling (26%):** legal Verus that encodes the wrong behavior, e.g., a wrong Entity–Model relation.
- **IV. Proof construction (14%):** near-correct property, but missing invariants, assertions, lemmas, or solver-usable facts.

Qwen-Coder fails late (Stages III–IV: **54%**); GPT-4o and DS-V4 fail early (Stages I–II: **62%** and **69%**).

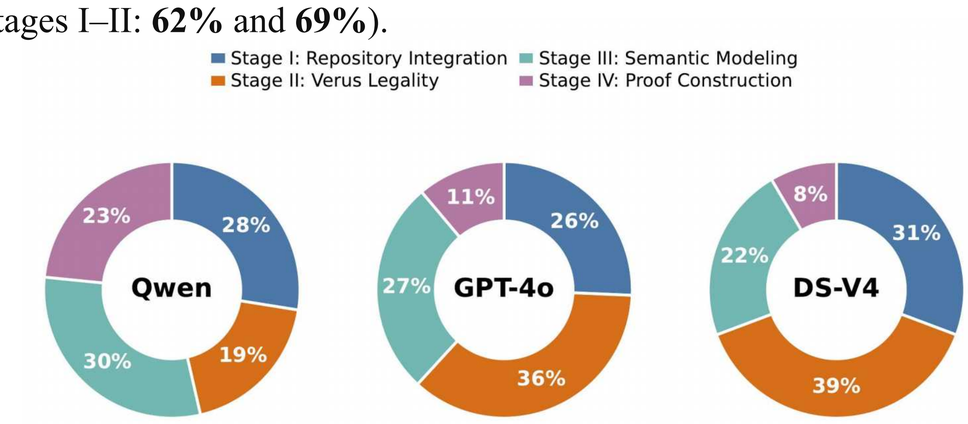


Fig. 5. Reviewed failures by stage and model

Finding 3: Diagnostics and Repair Are Unreliable

Diagnostics can mislead: in **27.4%** of episodes, the stage Verus reports is not where the defect actually lies.

Repair rarely converges: only 85 of 270 traces (**31.5%**) reach a Verus-confirmed pass; **15.1%** of episodes regress to compilation failures.

Better alignment does not imply better repair: GPT-4o aligns best (76.0%) yet converges least (26.5%).

Repair should first attribute the source stage, then apply stage-appropriate edits.

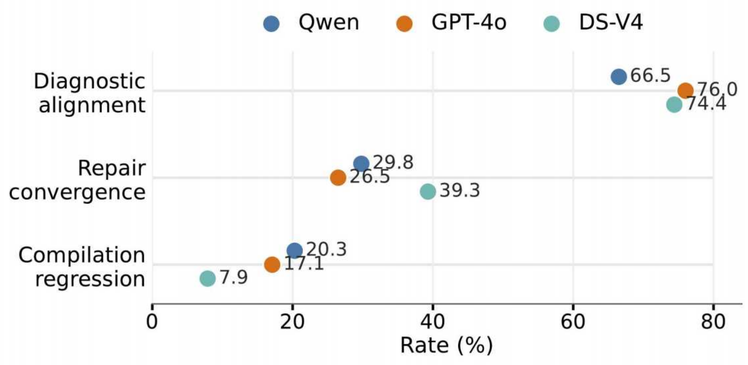


Fig. 6. Diagnostic alignment and repair reliability