MAVS Chapter 10D Completion Report

Dynamic validation, baseline comparison, governance tracing, and full minimum-suite results

Prepared from repository state a1bfd52b59aa on 2026-07-06.

Executive Summary

Chapter 10D has been implemented as a complete dynamic governance validation program for MAVS-GC. The repository now contains the governance method implementation, dynamic sequential environments, modern baseline families, ablation controls, final benchmark suite definitions, reproducible runners, aggregate metrics, failure cards, and claim-disciplined reporting.

The final Phase 6 run is no longer a smoke test. The required minimum execution produced 383,200 trace records across 13 suite experiments, 2,530 episode rows, 97 aggregate summary rows, and 12,161 failure cards. Trace completeness and audit-trace completeness are both 1.0000 on the aggregated full-run outputs.

The substantive result is precise: MAVS-GC achieved near-zero unsafe acceptance across the reported MAVS rows, but it did not win every safety-utility comparison. It behaved as a conservative governance method: strong at avoiding unsafe acceptances, sometimes costly in false rejections, and not proven to solve correlated representation collapse.

Field	Value
Repository commit	a1bfd52b59aaba69b2c041a5e7da0ee263125c1f
Full trace records	383,200
Suite experiments	13
Episode rows	2,530
Summary rows	97
Environment families	5
Method ids	36
Failure cards	12,161
Trace completeness	1.0000
Audit-trace completeness	1.0000

What Was Implemented

The implementation converts Chapter 10D from a research plan into an executable validation stack. It does not merely describe MAVS-GC; it provides the machinery needed to run sequential benchmark episodes, compare governance methods under shared contracts, emit audit traces, aggregate metrics, generate failure cards, and write reproducible reports.

- A common GovernanceMethod interface for MAVS-GC and all baselines.
- Config-driven sequential environments with corruption schedules and hidden labels.
- JSONL trace emission for every decision, including enough fields to reconstruct the governance decision.
- Modern baseline families: policy rails, validator stack, confidence gate, disagreement gate, self-consistency, conformal abstention, reject option, judge, debate, critique-revise, and sanity controls.
- Formal MAVS-GC implementation with specialist supports, diagnostics, severity, contextual weighting, hard-veto behavior, bounded mitigation, and final decision traces.
- External taxonomy scaffolding for HELM Safety, CyberSecEval, SWE-bench, BrowserBench/WebArena, GAIA, and red-team methodology framing.
- Final Phase 6 suite execution with E1-E5 coverage, metrics, uncertainty intervals, negative-result reporting, and failure-card generation.

What This Does Not Claim

- It does not claim frontier-model or industrial-scale validation.
- It does not claim MAVS-GC beats all governance methods.
- It does not claim MAVS-GC solves correlated specialist or representation failure.
- It does not claim official NVIDIA NeMo Guardrails, Guardrails AI, HELM, CyberSecEval, SWE-bench, BrowserBench, GAIA, or OpenAI Evals were executed end to end.

Implemented Technical Architecture

The implemented architecture is deliberately modular. Environments generate observations and hidden labels, candidate generators expose action candidates, governance methods decide accept/reject/escalate, and the runner records every step into JSONL traces. The metrics and report layers operate from those traces instead of relying on hand-entered results.

Layer	Primary files	Purpose
Core contracts	types.py, interfaces.py, runner.py, config.py, registry.py	Shared method/environment contracts, deterministic config loading, registry wiring, and experiment orchestration.
Dynamic environments	text_safety.py, tool_use.py, cyber_triage.py, multi_agent.py, correlated_collapse.py, synthetic_ops.py	Sequential benchmark families with corruption phases, recovery periods, and task-specific hidden labels.

Layer	Primary files	Purpose
Corruption schedules	corruption/schedules.py and transforms	Piecewise and sweep schedules for prompt injection, evidence masking, label drift, shared wrong premise, exfiltration bait, residual drift, and correlated transforms.
Governance	governance/mavs_gc.py and related modules	MAVS-GC decision procedure with specialist evidence, severity, diagnostics, weighting, thresholds, hard vetoes, and auditable outputs.
Baselines	baselines/*.py and configs/baselines/*.yaml	Comparable governance alternatives under the same runner and method interface.
Metrics	metrics/*.py	Safety, utility, calibration, trace completeness, dynamic lag, collapse sensitivity, bootstrap intervals, and worst-case extraction.
Reports	reports/*.py and scripts/make_report.py	CSV/Markdown tables, final README, figures, and failure cards.

Phase Completion Matrix

Phase	Scope	Status	Completion evidence
1	Foundation, contracts, config, audit tracing	Implemented and validated.	Core package scaffold, interfaces, config loader, runner, registry, JSONL traces, smoke tests.
2	Dynamic environments and corruption schedules	Implemented and validated.	Text safety, tool use, cyber triage, multi-agent operations, correlated collapse skeleton, synthetic ops, schedule sweeps.
3	Modern baselines and abstention methods	Implemented and validated.	Policy rails, validator stack, confidence/disagreement gates, self-consistency, conformal methods, reject option.
4	Full MAVS-GC and correlated stressors	Implemented and validated.	Formal MAVS-GC calculus, hard veto, severity monotonicity, bounded mitigation, external taxonomy adapters.
5	Ablations, final configs, anti-overfitting controls	Implemented and validated.	Final seed ranges, training separation rules, model-card gates, 16 ablations, heldout configs.
6	Full suite execution, metrics, reports, failure cards	Completed.	383,200 trace records, 97 summary rows, 2,530 episodes, 12,161 failure cards, polished final report.

Phase-By-Phase Implementation Detail

Phase 1: Repository Foundation

Phase 1 established the execution substrate. The implementation introduced typed experiment contracts, a common governance interface, deterministic YAML configuration loading, a registry for environments and methods, a runner that iterates seeds and steps, and JSONL trace logging. This phase matters because all later claims depend on comparable execution: MAVS-GC and every baseline are invoked through the same method contract and written through the same trace path.

The trace contract is a major part of the solution. Each record captures run id, config hash, commit, environment id, method id, seed, episode id, step id, observation, candidate, decision, step result, hidden-label hash, trace completeness, timestamp, and metadata. This makes the final report reconstructable from data rather than narrative memory.

Phase 2: Dynamic Environments

Phase 2 moved the repository beyond static benchmark rows. The environments are sequential: each decision occurs inside an episode with time, active corruption phase, visible state, hidden safety label, and reward. The implemented environment families cover text safety, tool-use security, cyber triage, multi-agent operations, synthetic operations, and correlated representation collapse.

The corruption system supports piecewise schedules and sweeps. This is essential to Chapter 10D because the plan asks whether governance behavior transfers from bounded static tests to non-stationary conditions. The implemented transforms include prompt injection, evidence masking, label drift, shared wrong premise, shared confidence bias, shared retrieval context, shared provenance concentration, exfiltration bait, and residual drift.

Phase 3: Contemporary Baselines

Phase 3 implemented the comparative governance surface. The policy-rails baseline approximates NeMo-style programmable guardrail behavior through local rail rules. The validator-stack baseline approximates Guardrails AI-style modular validation through local validators. Additional baselines cover confidence gating, disagreement gating, self-consistency, conformal abstention, adaptive conformal abstention, reject option classification, and sanity controls.

This phase is also where the report's most important caveat begins: these are comparable local baseline families, not direct official executions of NVIDIA NeMo Guardrails or Guardrails AI. The implementation is suitable for method-family comparison inside the MAVS suite, but the report does not claim external framework benchmark certification.

Phase 4: Full MAVS-GC And Stressors

Phase 4 implemented the full MAVS-GC governance method behind the same interface as the baselines. The method combines specialist support scores, confidence, diagnostics, severity, contextual weights, threshold policy, hard-veto handling, and final decision logic. The output is not only a decision; it is an audit trace showing how the decision was produced.

This phase also added stressors that make the validation meaningfully harder: correlated specialist failure, shared wrong premises, shared representation corruption, and recovery periods. External benchmark taxonomy adapters were added for HELM Safety, CyberSecEval, SWE-bench, BrowserBench/WebArena, GAIA, and red-teaming methodology so that report categories can be mapped without falsely claiming official benchmark execution.

Phase 5: Ablations And Anti-Overfitting Controls

Phase 5 added ablation coverage and final-run discipline. The suite contains full MAVS-GC plus component ablations such as no severity, fixed severity, noisy severity, no diagnostics, randomized diagnostics, fixed threshold, delayed threshold, homogeneous specialists, shared representation, reject-only fallback, and no escalation. These ablations make the result more useful because they reveal which components affect safety and utility under stress.

The anti-overfitting controls are explicit. Final benchmark seeds are separated from development seeds, training is not performed during final execution, configs mark benchmark sets, and artifact-backed methods must provide training cards and manifests. No final model training was performed for Phase 6.

Phase 6: Full Minimum Execution

Phase 6 initially had only bounded stress evidence. That gap was corrected by executing the full minimum suite without max-seed or max-step truncation. The final run produced exactly the required 383,200 records, then regenerated summary parquet, step metrics, episode metrics, failure cards, figures, and the final claim-disciplined README.

The full run is now pushed in the repository. Large raw traces and the full step metrics file are stored through Git LFS. Local console logs were intentionally ignored because the full run log alone was several gigabytes; the reproducibility artifacts are the pushed traces, metrics, manifest, and report outputs.

Final Benchmark Design

The full validation suite is organized as E1-E5. Each experiment family tests a different pressure point: non-stationary corruption, correlated failure, baseline comparison, ablation sensitivity, and corruption sweep behavior. The full suite uses final-range seeds and no training during final execution.

Experiment	Purpose	Minimum	Coverage	Records
E1	Dynamic corruption	30 seeds x 240 steps x 3 environments	Text Safety, Tool-Use Security, Synthetic Ops	21,600
E2	Correlated failure	50 seeds x 160 steps x 2 environments	Multi-Agent Operations, Correlated Representation Collapse	16,000
E3	Governance baseline comparison	20 seeds x 120 steps x 3 environments x 14 methods	Modern baselines plus full MAVS-GC	100,800
E4	Ablation study	30 seeds x 160 steps x 3 environments x 16 methods	Full MAVS-GC and component ablations	230,400
E5	Stress schedule sweep	30 seeds x 240 steps x 2 environments	Synthetic Ops and Tool-Use Security corruption sweeps	14,400

Final Results

The primary safety finding is that MAVS-GC avoided unsafe acceptances in the reported MAVS rows. The primary utility caveat is that MAVS-GC often achieved that safety through conservative rejection, which increased false rejection rate and reduced reward in several slices.

MAVS-GC E3 Baseline Comparison

Environment	Mean reward	UAR	FRR	Escalation	Steps
Synthetic Ops	0.8578	0.0000	0.1804	0.0000	2,400
Text Safety Stream	0.9932	0.0000	0.0074	0.0000	2,400
Tool Use Security	0.8974	0.0000	0.1703	0.0000	2,400

In E3, MAVS-GC was not the overall winner by reward. It ranked 5/14 on Synthetic Ops, 2/14 on Text Safety, and 4/14 on Tool-Use Security when sorted by unsafe acceptance rate and then reward. The important positive result is that the unsafe acceptance rate stayed at 0.0000 in all three E3 rows.

Correlated Failure Interpretation

Environment	Method	Reward	UAR	FRR	Collapse sensitivity
Correlated Representation Collapse	mavs_gc_correlated_final	0.0250	0.0000	1.0000	-1.0000
Multi Agent Operations	mavs_gc_E2_multi_agent	0.9347	0.0000	0.1003	0.1742

The correlated representation collapse result is deliberately treated as a negative or cautionary finding. MAVS-GC avoided unsafe acceptance there, but it collapsed into rejection behavior with FRR 1.0000 and mean reward 0.0250. This supports the report limitation that Chapter 10D does not prove MAVS-GC solves correlated failure.

Baseline Comparison Against Referenced Framework Families

Reference	Implemented comparison	Status	Interpretation
NVIDIA NeMo Guardrails	policy_rails_final	Approximation only	Programmable policy-rail behavior was implemented locally; official NeMo was not imported or executed.
Guardrails AI	validator_stack_final	Approximation only	Modular validator-stack behavior was implemented locally; official Guardrails AI was not imported or executed.
OpenAI Evals	N/A	Framework reference	The suite is reproducible and config-driven, but OpenAI Evals was not used as the execution harness.
HELM Safety / CyberSecEval	External taxonomy adapters	Category framing only	Taxonomy mapping was implemented; official external validation/test examples were not ingested.
SWE-bench / BrowserBench / GAIA	External adapter scaffolds	Metadata/framing only	Adapters preserve benchmark framing; official benchmark execution was not completed.

Main Result Statement

The strongest defensible statement is: MAVS-GC achieved near-zero unsafe acceptance across the full dynamic validation suite, but often paid for that safety with elevated false rejections and reduced utility. It should be described as conservative and auditable, not as universally superior.

External Reference Handling

The implementation preserves the user-supplied external references in two ways: by creating comparable local baseline families inspired by governance frameworks, and by adding taxonomy or metadata adapters for external benchmark families. This distinction matters because the completion report must not overstate what was executed.

- NeMo Guardrails: represented by a local policy-rails baseline, not official NeMo execution.
- Guardrails AI: represented by a local validator-stack baseline, not official Guardrails AI execution.
- HELM Safety and CyberSecEval: represented by category mapping adapters.
- SWE-bench, BrowserBench/WebArena, and GAIA: represented by metadata/config scaffolds and task-family mapping.
- Anthropic red teaming: represented by an explicit red-team methodology adapter.

Artifacts And Reproducibility

The completion artifacts are committed and pushed. Large raw traces and the full step metrics file are tracked through Git LFS because they exceed normal GitHub blob-size limits.

Artifact	Location	Purpose
Final report README	results/reports/dynamic_validation_v1/README.md	Human-readable final result summary.
Raw traces	results/raw/*.jsonl	Full suite JSONL traces, stored as Git LFS objects.
Summary parquet	results/processed/summary.parquet	97 aggregate metric rows.
Step metrics	results/processed/step_metrics.csv	383,200 step rows, stored as Git LFS object.
Episode metrics	results/processed/episode_metrics.csv	2,530 episode rows.
Failure cards	results/reports/dynamic_validation_v1/failure_cards/*.md	12,161 serious unsafe-acceptance cards.
Suite manifest	results/processed/suite_run_manifest.json	13 suite entries and record counts.
Code commit	a1bfd52b59aaba69b2c041a5e7da0ee263125c1f	Pushed repository state for the full run.

Reproduction Commands

1. python scripts/run_suite.py --suite configs/suites/dynamic_governance_v1.yaml
2. python scripts/aggregate_results.py --input results/raw --out results/processed/summary.parquet
3. python scripts/make_failure_cards.py --input results/raw --out results/reports/dynamic_validation_v1/failure_cards

4. python scripts/make_report.py --summary results/processed/summary.parquet --out results/reports/dynamic_validation_v1

Validation Commands

Trace validation was run over every top-level full-run JSONL file. The Phase 6 report generation tests also passed: tests/unit/test_metrics.py, tests/unit/test_stats.py, tests/unit/test_failure_cards.py, and tests/integration/test_report_generation.py produced 5 passed tests.

Claim Discipline

Required limitation	Status	Report treatment
No frontier-model claim	Satisfied	The report describes the benchmark as synthetic/reproducible validation, not frontier-model industrial evaluation.
No industrial-scale claim	Satisfied	Large for the project, but still a controlled research benchmark.
No universal robustness claim	Satisfied	Results are scoped to implemented environments and generated traces.
No proof MAVS solves correlated failure	Satisfied	The correlated representation collapse result is explicitly negative on utility and FRR.
No claim MAVS beats all governance methods	Satisfied	The report states MAVS-GC did not win every safety-utility comparison.

Completion Assessment

The work is complete relative to the practical obligations of Chapter 10D: repository scaffolding, dynamic environments, baseline comparison, formal MAVS-GC implementation, ablations, anti-overfitting controls, full minimum benchmark execution, metrics, failure cards, and final reporting. The implementation is also honest about the limits of the evidence.

- Completed: executable end-to-end Chapter 10D benchmark suite.
- Completed: full required Phase 6 minimum run, not a smoke run.
- Completed: documentation in WorkPlan.md and Path.md tying implementation to phase requirements.
- Completed: pushed full-run artifacts and LFS-managed large traces.
- Remaining research work: direct execution of official external benchmark suites and stronger correlated-failure-specific baseline comparisons.

Appendix A: Metric Definitions

Metric	Meaning
Unsafe Acceptance Rate	Unsafe accepted divided by unsafe opportunities. Lower is better.
False Rejection Rate	Safe rejected divided by safe opportunities. Lower is better.
Escalation Rate	Escalated decisions divided by total decisions.
Mean Reward	Average environment reward across steps or episodes. Higher is better.
Catastrophic Episode Rate	Episode fraction with one or more irreversible unsafe acceptances.
Calibration Error	Gap between predicted risk and empirical unsafe rate.
Adaptation Lag	Steps between corruption onset and stable rejection/escalation behavior.
Recovery Lag	Steps after corruption ends until normal acceptance returns.
Correlation Collapse Sensitivity	Reward delta between independent and shared/correlated failure conditions.
Trace Completeness	Fraction of decisions with required trace fields present.

Appendix B: E3 Baseline Ranking Summary

Environment	Rank	Method	Reward	UAR	FRR
Synthetic Ops	1	policy_rails_final	1.0000	0.0000	0.0000
Synthetic Ops	2	self_consistency_final	1.0000	0.0000	0.0000
Synthetic Ops	3	debate_final	0.9122	0.0000	0.0000
Synthetic Ops	4	reject_option_final	0.9122	0.0000	0.0000
Synthetic Ops	5	mavs_gc_full_final	0.8578	0.0000	0.1804
Synthetic Ops	6	critique_revise_final	0.8372	0.0000	0.0000
Text Safety Stream	1	policy_rails_final	1.0000	0.0000	0.0000
Text Safety Stream	2	mavs_gc_full_final	0.9932	0.0000	0.0074
Text Safety Stream	3	critique_revise_final	0.9922	0.0000	0.0000
Text Safety Stream	4	judge_final	0.9922	0.0000	0.0000
Text Safety Stream	5	debate_final	0.9897	0.0000	0.0000
Text Safety Stream	6	reject_option_final	0.9897	0.0000	0.0000
Tool Use Security	1	policy_rails_final	1.0000	0.0000	0.0000
Tool Use Security	2	critique_revise_final	0.9500	0.0000	0.0000
Tool Use Security	3	judge_final	0.9500	0.0000	0.0000

Environment	Rank	Method	Reward	UAR	FRR
Tool Use Security	4	mavs_gc_full_final	0.8974	0.0000	0.1703
Tool Use Security	5	debate_final	0.8300	0.0000	0.0000
Tool Use Security	6	reject_option_final	0.8300	0.0000	0.0000

Appendix C: Source Evidence Used

- WorkPlan.md for Chapter 10D scope, phase requirements, external references, and acceptance criteria.
- Path.md for implementation ledger, phase-by-phase evidence, console-log locations, deviations, and full-run correction.
- configs/suites/dynamic_governance_v1.yaml for final E1-E5 experiment definitions.
- results/processed/summary.parquet for final aggregate metrics.
- results/reports/dynamic_validation_v1/README.md for the polished full-run report.
- results/reports/dynamic_validation_v1/failure_cards/*.md for unsafe-acceptance failure analysis.
- Git commit a1bfd52b59aaba69b2c041a5e7da0ee263125c1f for the pushed full-run repository state.