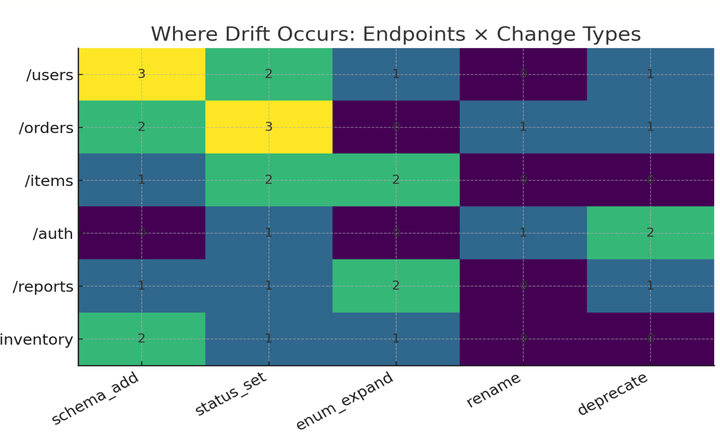
TITLE: Agentic, Evidence-Backed Maintenance for Resilient API Checks

Name: Vijeth Ganapatigouda Patil, Information Technology(MS)
Mentor: Brian Atkinson, Assistant Teaching Professor
The Polytechnic School

Research question: Can we cut API test maintenance time by automatically diagnosing failures and proposing safe, minimal patches—without auto-committing or risking regressions?
Motivation. API suites are brittle: minor, non-breaking changes (201 vs 200, optional fields, header casing) trigger red failures. The cost is reviewer time and trust. Our target is a loop that (1) localizes the fault, (2) proposes a scoped fix, (3) proves it safe in a sandbox, and (4) surfaces a human-readable report/PR—no auto-merge.

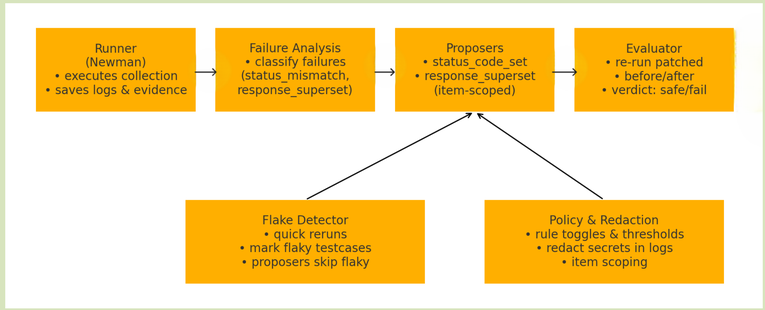


Technical Highlights

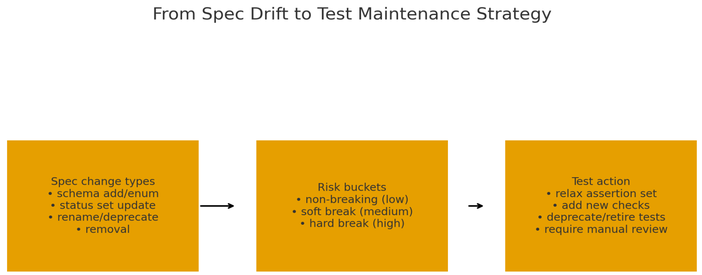
Item-scoped rewriting using source.name from failure evidence prevents global over-relaxation—e.g., we no longer make slideshow optional on endpoints where it's legitimately required.
Deterministic rule pack vs. "AI auto-edit." Patches are mechanistic and auditable; the Evaluator is the trusted oracle.
Cross-platform executor (Windows/Mac/Linux): robust npx/newman discovery; no shell-specific tricks.
Encoding resilience: BOM-tolerant JSON reads and CP-1252 decoding fallback eliminated parsing failures on Windows logs.
Security & compliance: recursive redaction for strings and nested structures ensures artifacts can be shared/attached in PRs.

Methods — Deterministic Agentic Loop

Pipeline: Runner → Failure Analysis → Proposers → Evaluator → Report, with Flake Detection and Policy/Redaction as guardrails.

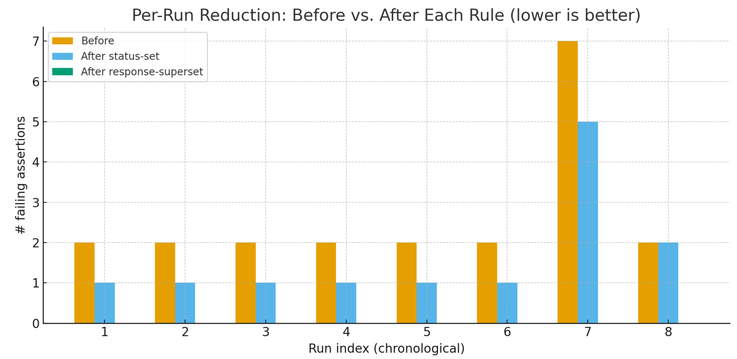


Runner (Newman): Executes Postman collections, outputs structured JSON and logs, and stores per-failure evidence.
Failure Analysis: Classifies failure causes (e.g., status mismatches or response superset issues) using heuristics.
Proposers: Generate targeted item-level patches with diffs and rationale, adjusting status codes or property checks based on failure type.
Evaluator: Re-runs patched collections in a sandbox to verify safety and measure improvement.
Report: Summarizes failures, proposals, and evaluations with overall impact metrics.
Guardrails: Detect and skip flaky tests, enforce policy thresholds, and redact sensitive data from all logs and evidence.



Findings

Impact: Across 8 shared runs, the best patch reduced failures from 2→0 in the stable sample, and 7→0 in a noisier collection after item-scoping and parse-guards.
Evaluator gates safety: In every run with a proposer, at least one patch evaluated safe with no new failures introduced.
The dominant classes observed were response_superset and status_mismatch; only rare "other" errors
Noise removal: Guarding pm.response.json() prevented pre-assertion throws; flake detection labeled nondeterministic status endpoints as flaky, removing them from proposal scope.



Future Scope

Automate safe patch handling via draft PRs with reports and rule-based labels.
Enable faster, targeted re-evaluations using Newman's folder-level runs.
Correlate failures with OpenAPI spec changes to flag possible spec drift.
Use LLMs for fix pattern prediction, root-cause explanations, and policy-gated auto-resolves in stable repositories.