VIBE CODING FAILURE PATTERNS AND SOLUTIONS

A Field Guide for Vibe Coding Disasters and Recovery

From the Book *Vibe Coding: Building Production-Grade Software With GenAI, Chat, Agents, and Beyond* by Gene Kim and Steve Yegge, Published by IT Revolution, 2025.

AI coding assistants can generate code faster than humans can write it, but this speed comes with new categories of failure that didn't exist in traditional development. This guide catalogs the most common disaster patterns that emerge when working with AI across the three developer loops described in *Vibe Coding*—from immediate coding mistakes that can be caught in seconds to architectural catastrophes that can destroy weeks of work.

Each pattern includes the warning signs to watch for, immediate response procedures to limit damage, and prevention strategies to avoid the problem entirely. The patterns are organized by timeframe: Inner Loop failures happen in seconds to minutes during active AI collaboration, Middle Loop failures emerge over hours to days as context degrades and multiple agents interact, and Outer Loop failures develop over weeks to months when architectural boundaries break down. Use this as both a diagnostic tool when something feels wrong and an emergency response manual when disasters strike.

UNIVERSAL EMERGENCY PROCEDURES:

When in doubt:

1. STOP all AI agents immediately
2. Assess which loop the problem belongs to
3. Check version control status
4. Create backup before attempting fixes
5. Start with simplest solution first
6. Document the incident for future prevention

INNER LOOP FAILURES (Seconds to Minutes)

"TESTS ARE PASSING" LIE — High

Symptoms:
- Claims "All tests green" but code doesn't compile
- Claims tests exist when none were written
- Reports success after making changes that break everything

Immediate Solution:
- Run tests yourself independently
- Have AI demonstrate by running tests in front of you
- Never commit based on AI claims alone

Prevention:
- Set up automatic test running on file save
- Make "show me the test output" your default response

CONTEXT AMNESIA — Medium

Symptoms:
- AI forgets instructions from 5 minutes ago
- Repeats same mistakes after corrections
- Ignores project-specific rules

Immediate Solution:
- Stop and quiz AI: "What are you working on?"
- Re-state critical constraints
- Clear context and start fresh if necessary

Prevention:
- Monitor context window usage
- Keep tasks small and focused
- Proactively clear context at 50% capacity

INSTRUCTION DRIFT — Medium

Symptoms:
- AI gradually deviates from original requirements
- Starts "improving" things you didn't ask for
- Adds features beyond scope

Immediate Solution:
- Interrupt and redirect: "Stop. Focus only on X"
- Re-read original specification together
- Use smaller, more specific tasks

Prevention:
- Write explicit acceptance criteria
- Use "surgical" task definitions
- Regular check-ins: "Are we still on track?"

DEBUG LOOP SPIRAL — High

Symptoms:
- AI adds more logging instead of fixing root cause
- Creates increasingly complex debug attempts
- Floods context with verbose logging output

Immediate Solution:
- Take manual control of debugging
- Use actual debugger instead of print statements
- Start new session with simplified problem

Prevention:
- Set debug attempt limits
- Prefer debugger over logging
- Use "tracer bullet" approach for complex problems

Security Levels

Critical Can destroy code/data — *High* Breaks functionality

Medium Reduces quality — *Low* Minor annoyance

MIDDLE LOOP FAILURES (Hours to Days)

ELDRITCH CODE HORROR — Critical

Symptoms:
- 3,000+ line functions that "work" but are incomprehensible
- Everything connected to everything else
- Changing one line breaks distant components

Immediate Solution:
- Stop all feature work immediately
- Implement comprehensive tests before refactoring
- Modularize in small, tested increments

Prevention:
- Enforce modular boundaries from day one
- Regular architecture reviews
- "Code elegance" checks after each AI session

EMERGENCY PROTOCOL:
- If you can't understand code AI wrote, stop everything. This technical debt that will compound exponentially.

AGENT WORKSPACE COLLISION — High

Symptoms:
- Multiple agents modifying same files
- Port conflicts from parallel services
- Agents working in wrong directories/branches

Immediate Solution:
- Pause all agents and assess workspace state
- Manually resolve conflicts before continuing
- Reassign clear, non-overlapping territories

Prevention:
- Color-code terminal windows by workspace
- Use different Git branches per agent
- Maintain agent coordination document

MEMORY TATTOO DECAY — Medium

Symptoms:
- AI can't resume previous work effectively
- Keeps re-solving already-solved problems
- Progress documents become stale or incorrect

Immediate Solution:
- Create fresh specification from current state
- Clean up outdated documentation
- Re-establish context with working examples

Prevention:
- Regular "tattoo" maintenance
- Date-stamp all progress documents
- Delete plans immediately when completed

MULTI-AGENT DEADLOCK — High

Symptoms:
- Agents waiting for each other's output
- Circular dependencies in task assignments
- No agent can make progress

Immediate Solution:
- Manually break the dependency cycle
- Assign 1 agent as "primary" for shared resources
- Sequence dependent tasks explicitly

Prevention:
- Map task dependencies before assignment
- Use "tracer bullet" approach for integration
- Regular coordination check-ins

Security Levels

Critical Can destroy code/data — *High* Breaks functionality

Medium Reduces quality — *Low* Minor annoyance

OUTER LOOP FAILURES (Weeks to Months)

BRIDGE TORCHING (API BREAKAGE) — Critical

Symptoms:
- AI changes function signatures without warning
- Removes or renames APIs other code depends on
- "Improvements" that break existing functionality

Immediate Solution:
- Roll back changes immediately
- Audit all API dependencies
- Implement version compatibility layer

Prevention:
- Golden rule: "You cannot break existing functionality"
- API compatibility tests in CI/CD
- Accretion-only development philosophy

EMERGENCY PROTOCOL:
- API breakage = customer impact. Prioritize rollback over fix-forward unless damage is already done.

REPOSITORY DELETION DISASTER — Critical

Symptoms:
- AI deletes "unused" branches containing work
- Whole directories disappear
- Git history becomes corrupted or lost

Immediate Solution:
- STOP - don't make any more Git operations
- Check Git reflog for recovery options
- Use AI to perform "Git surgery" recovery

Prevention:
- Push to remote frequently
- Never let AI delete branches without manual approval
- Regular backup verification

EMERGENCY PROTOCOL:
- Emergency recovery commands:

```
git reflog --all git fsck --lost-found
git show-branch --all
```

ORGANIZATIONAL PROCESS GRIDLOCK — High

Symptoms:
- AI productivity gains negated by approval processes
- Multiple manual review gates
- Deploy times measured in weeks

Immediate Solution:
- Create "fast lane" for low-risk changes
- Automate security and compliance checks
- Use data to demonstrate safety of smaller deployments

Prevention:
- Invest in automated testing/security
- Build organizational AI readiness
- Train AI to work within existing constraints

STEWNAMI (WORKSPACE CHAOS) — High

Symptoms:
- Changes in one area break completely unrelated features
- Impossible to track which agent made which changes
- Merge conflicts spanning hundreds of files

Immediate Solution:
- Freeze all agent activity
- Use AI to perform heroic merge recovery
- Restructure workspace boundaries before resuming

Prevention:
- Clear workspace partitioning from the start
- Regular branch hygiene
- Coordination documentation

Security Levels

Critical Can destroy code/data — *High* Breaks functionality

Medium Reduces quality — *Low* Minor annoyance