Table of Contents

1 Development Philosophy: Iterative Delivery at Scale

1.1 Executive Summary

This document outlines a development philosophy centered on **disciplined iteration**, **semantic versioning**, and **team coordination**. Rather than theoretical principles, it presents evidence from 3.5+ years of sustained GitHub activity demonstrating production-oriented software development.

The core thesis: **small, frequent releases with automated versioning reduce coordination overhead and deployment risk**—directly applicable to Strata Space's July 2026 milestone with 3 parallel squads (December 2025 complete).

1.2 Release Discipline: Evidence from Production

1.2.1 Project History

Project	Duration	Commits	Releases	Pattern
rsr-fsa	3.5+ years	34	-	Long-term simulation
claude-code	10 months	375	-	CLI tooling, 288 active days
scripts	6 months	205	-	macOS automation ecosystem
cc-skills	2+ months	955+	150+	18 plugins, 100+ skills (industry standard)
netstrata	27 days	118	34	Responsive iteration (1.3/day)
gapless-crypto	73 days	187	10	PyPI published, CI/CD automated
alpha-forge	42 days	81	-	42 ADRs (architecture-first)

1.2.2 Key Observations

1. **Sustained activity, not recent burst**: The oldest project (rsr-fsa) dates to March 2022. This represents 3.5+ years of continuous development across financial modeling, automation, and data engineering domains.
2. **Commit discipline**: In the netstrata project, 127 of 132 commits occurred on unique dates (96.2%)—evidence of daily, disciplined work rather than bulk commits.

3. **Release velocity scales with need**: cc-skills achieved 150+ releases over 2+ months with sustained intensity (955+ commits), while gapless-crypto maintained 10 releases over 73 days for a production PyPI package. The methodology adapts to project phase.
4. **Industry standard adoption**: cc-skills implements Agent Skills—now adopted by 8+ major AI coding tools including Claude Code, GitHub Copilot, Cursor, and OpenAI Codex. Skills transfer across tools—no vendor lock-in.

1.3 Commit Hygiene Examples

All projects follow Conventional Commits with semantic-release automation:

```
feat(skills): add clickhouse-architect skill for schema design
fix(validation): correct ALP codec recommendation per ClickHouse docs
docs(adr): link release notes to architecture decisions
chore(release): 2.30.0 [skip ci]
refactor(config): centralize environment variables via mise [env]
```

Why this matters for Strata Space: When 3 squads work in parallel, consistent commit conventions enable:

- Automated changelog generation (no manual release notes)
- Clear attribution of changes across modules (DMS vs Tasks vs Time Recording)
- Version tracking that supports staged rollouts

1.3.1 Git Workflow Enforcement

Beyond commit conventions, git-town-workflow plugin blocks raw git commands requiring standardized equivalents:

Blocked Command	Use Instead
`git checkout -b`	`git town hack`
`git pull`	`git town sync`
`git merge`	`git town sync`
`git rebase`	`git town sync`

Safe operations allowed: `git add`, `git commit`, `git status`, `git log`, `git diff`

Why this matters for Strata Space: 8 new Manila developers + 3 squads follow consistent branching patterns from day 1. Prevents "cowboy coding" that can emerge when teams scale rapidly.

1.4 Architecture Decision Records (ADRs)

1.4.1 Evidence: 42 ADRs in alpha-forge

The alpha-forge project maintains 42 formal Architecture Decision Records documenting technical trade-offs. Sample titles:

- `2025-11-17-e2e-first-testing-strategy.md` — Quantified analysis: 2.6:1 maintenance-to-implementation ratio
- `2025-11-14-dsl-simplification-v04.md` — Breaking change management with migration path
- `2025-11-17-plugin-metadata-management.md` — Ecosystem coordination patterns

1.4.2 Why ADRs Support Squad Coordination

ADRs create **decision traceability** that benefits:

1. **Onboarding**: New team members understand *why* decisions were made, not just *what* was built
2. **Squad alignment**: When Squad A's decision affects Squad B's module, the rationale is documented
3. **Future refactoring**: Six months later, the team knows whether to preserve or change a pattern

1.4.3 Session Recording for Context Capture

ADRs document decisions; asciinema-tools captures the exploration that led to them:

- **950:1 compression**: 3.8GB terminal session → 4MB searchable text
- **Onboarding use**: "Watch how I debugged the DMS migration issue" with full terminal context
- **Streaming backup**: Real-time backup to GitHub orphan branch (no lost sessions)

Why this matters for Strata Space: When senior developers solve complex issues, capture the full context for future reference. ADRs alone cannot convey the debugging process—session recordings can.

1.4.4 Terminology Consistency

Vale glossary management enforces consistent terminology as Single Source of Truth (SSoT):

- 3 squads use same terms: "lot owner" vs "unit holder", "strata plan" vs "scheme", "by-law" vs "bylaw"
- New developers adopt correct terminology automatically
- Hooks detect when different CLAUDE.md files define terms inconsistently

1.4.5 Sample ADR Structure (MADR 4.0)

```
---
status: accepted
date: 2025-11-17
decision-maker: terrylica
consulted: [team members, Claude Code analysis]
research-method: Web research + codebase metrics
---

## Context and Problem Statement
[Quantified problem with metrics]

## Decision Drivers
[Prioritized criteria]

## Considered Options
[Option A, B, C with trade-offs]

## Decision Outcome
[Chosen option with rationale]
```

1.5 Application to Strata Space

1.5.1 December 2025 Milestone (Complete)

Challenge: 3 squads delivering DMS, Tasks, and Time Recording simultaneously to 120+ internal users.

How iterative release discipline helps:

- Each squad releases independently when their module is ready
- Semantic versioning tracks which features are in which build
- Automated changelogs reduce Tom's coordination overhead
- Rollback granularity: if Tasks has issues, DMS/Time Recording continue

Status: December 2025 delivered. Post-crunch stabilization underway.

1.5.2 July 2026 Milestone

Challenge: Buildings, Asset Management, Strata Manager Dashboard—targeting 90-95% of workload in Strata Space.

How ADR discipline helps:

- Architecture decisions documented before implementation
- Breaking changes planned with migration paths
- New squad members onboard faster with decision context
- Compliance documentation generated from release history

1.5.3 Reducing "Big Bang" Risk

Ted's observation: *"120 people, hey, there's a problem here"* when encountering screen changes.

Iterative approach reduces this risk by:

- Validating each module with a subset of users before company-wide rollout
- Catching integration issues between squads early (not at final merge)
- Providing rollback points if adoption issues emerge

Code Quality Enforcement:

itp-hooks catches silent failures before they reach production:

- Bare `except:` (hides KeyboardInterrupt)
- `subprocess` without `check=True` (silent failures)
- NOT cosmetic issues (unused imports, PEP8 style)

This addresses Ted's concern about AI producing "garbage"—every commit is validated for runtime correctness, not stylistic preferences.

Cross-Module Integration Validation:

Symmetric dogfooding validates modules against each other before release:

- DMS module exports → Buildings module imports (and vice versa)
- December release validated against July development
- Squad A changes don't break Squad B's consumption

Example for Strata Space: Before releasing DMS changes, run integration tests against Buildings module. If Buildings has pending changes, test against their feature branch too. Catches cross-squad integration failures before production.

1.6 Verify on GitHub

Profile: github.com/terrylica

Each repository’s commit history is publicly visible. Notable repositories for verification:

Repository	What to Verify
cc-skills	64 releases in 9 days, semantic-release automation
gapless-crypto-clickhouse	PyPI publishing, GitHub Actions CI/CD
netstrata	34 releases, responsive iteration to feedback

1.7 Summary

This development philosophy is not theoretical—it’s demonstrated through:

- **3.5+ years** of sustained GitHub activity
- **955+ commits** in cc-skills ecosystem (18 plugins, 100+ skills)
- **42 ADRs** documenting architectural decisions
- **96.2% unique commit dates** showing disciplined daily work
- **Industry standard adoption**: Agent Skills work across 8+ AI coding tools

For Strata Space, this translates to:

- Reduced coordination overhead for Tom’s 3 parallel squads
- Clear version tracking for July 2026 staged rollout (December 2025 complete)
- Decision traceability that accelerates onboarding
- Risk mitigation through smaller, more frequent releases
- Cross-module validation before releases (symmetric dogfooding)
- Consistent terminology across squads (Vale glossary management)