Contents

1 AI-Augmented Development Methodology

1.1 Introduction: State of the Art Meets Production Reality

Modern software development has reached an inflection point. AI coding agents—particularly Anthropic's Claude Code CLI and similar tools—have transitioned from experimental curiosities to production-grade development accelerators. However, the gap between casual experimentation and effective production usage is measured in extensive, sustained practice, not blog posts written after weekend trials.

This document outlines my approach to AI-augmented development workflows, the economic rationale behind systematic tooling adoption, and the professional requirements necessary for these methodologies to succeed in team environments.

1.2 Depth of Practice

1.2.1 Investment Profile

1000+ hours of hands-on practice across multiple AI coding platforms:

- **Claude Code CLI**: Extensive Agent Skills development, custom workflow design, and adoption advocacy
  - **Usage attestation**: 60,374+ sessions since July 2024 (https://share.cleanshot.com/xN1DCR9X)
  - 528 million tokens processed, $2,953 API costs demonstrating systematic production usage
- **Production deployment**: Not prototyping or experimentation—these tools run critical business workflows daily
- **Cross-platform experience**: Claude Code, Codex, Cursor, and other AI-augmented IDEs

- **Production systems built**: Multiple business-critical systems developed entirely using AI-augmented workflows

What extended practice reveals: 1000+ hours of production usage uncovers capabilities and limitations that aren't apparent in initial exploration. Patterns emerge about what works consistently, what fails unpredictably, and how to structure workflows for reliable results.

1.2.2 Claude Code CLI Specialization

Agent Skills Development: Active contribution to Claude Code ecosystem through:

- Custom Agent Skills creation for specialized workflows
- Internal popularization at Eon Labs (team adoption and training)
- Deep understanding of Claude Code's tool use patterns, context window management, and autonomous agent capabilities
- Experience with both successful and failed automation attempts—understanding each AI model's capability boundaries through iterative audit and constraint refinement

What "mastery" actually means:

- Understanding prompt engineering patterns that produce consistent results
- Structuring prompts to delegate all implementation decisions to AI agents while maintaining human oversight on business requirements
- Designing context-aware workflows that leverage context window rollover (i.e. auto-compacting) effectively
- Building reliable systems on top of non-deterministic AI outputs
- Managing context window usage through efficient prompt design

1.2.3 My Practice: AI-First Development Methodology

Through extensive production usage, my workflow has evolved to maximize AI agent capabilities while focusing human effort where it delivers unique value.

Human Role—Bridging Business Reality to AI Implementation:

- **Business requirements interpretation**: Understanding real-world needs and translating them into clear specifications
- **Architecture selection**: Choosing appropriate architectural patterns based on business constraints, scalability needs, and team capabilities
- **Prompt engineering**: Crafting precise, thorough prompts that communicate requirements and context to AI agents

- **Output review and iteration**: Auditing AI-generated implementations, identifying issues, refining prompts based on results

AI Role—All Discretionary Implementation Decisions:

In my practice, AI agents handle:

- Complete implementation (code generation, testing, documentation)
- Architecture execution and detailed design decisions
- Requirements analysis and technical specification
- Code auditing and quality assessment
- Refactoring and optimization decisions

Key Principle: Humans make business-driven decisions (what to build, why it matters, which architecture fits business needs). AI makes all discretionary technical decisions (how to implement, which libraries, specific patterns, code structure).

Why This Works: The interface between business reality and AI capability is human prompting. My focus is making that bridge as accurate and thorough as possible—not second-guessing technical implementation details that AI agents handle more consistently at scale.

Important Clarification: This describes my evolved methodology after extensive practice. It's not a universal prescription. Teams adopting AI-augmented development typically progress gradually, and different practitioners find different balances based on their experience level and comfort with agent autonomy.

1.3 Core Principle: Let AI Agents Use Standard Patterns

Business advantage comes from proprietary knowledge (NSW regulatory expertise, operational workflows). **Software robustness comes from standard patterns** (widely-used libraries, community-validated architectures).

Through extensive production usage, I've learned to let AI agents handle all discretionary technical decisions using idiomatic, modern patterns—while reserving human judgment for business requirements and domain-specific NSW regulatory knowledge.

Practical approach:

- Start with minimal constraints, audit outputs systematically
- Let agents discover current best practices rather than prescribing outdated approaches
- Add constraints only where audit reveals actual problems, not preemptively

- Trust agents for routine implementation, reserve human judgment for architecture and NSW compliance

For Netstrata's $12-14M software investment: Competitive advantage comes from the decade of NSW strata management expertise embedded in the system, not from custom technical patterns. Using standard, well-supported tools maximizes maintainability and protects that investment.

1.4 Non-Overlapping AI Contributions

Critical Context: Netstrata already has an AI program led by Epi Neto (IT Administrator, Masters in AI). My AI expertise is in a **different domain** that doesn't compete with or overlap his initiatives.

Netstrata's AI Program (Epi Neto's domain - business automation):

- **AI Training & Knowledge Bot** (pilot underway, showcase end Jan 2026): Internal chatbot for operational/legislative queries, drawing from Netstrata training materials, NSW Fair Trading, and NCAT sources [Tom Q3 Report, p.16]
- **Fire Asset Quote Pricing Comparison**: AI-driven analysis for competitive quotes
- **Meeting Minutes Action Extraction**: Automated extraction of actions from meeting minutes and compliance documentation [Tom Q3 Report, p.16] — Tom's highest-priority use case for "efficiency and compliance across broader range of business"

These use cases require deep understanding of Netstrata's operations, NSW regulations, and business context—Epi's domain expertise. Sydney (Business Analyst from the Strata Space team) has joined to assist Epi in project managing the AI program [Tom Q3 Report, p.12].

My AI Expertise (development tooling and practice):

- AI coding assistants (Claude Code CLI, Cursor) for rapid implementation
- Production automation workflows using AI agents
- Standard pattern generation and testing
- Prompt engineering for software development velocity

Why These Don't Overlap: Business automation (chatbots, document processing, pricing analysis) requires deep company knowledge and regulatory context. Development tooling accelerates software delivery through AI-assisted coding. These are **complementary domains**, not competing efforts.

Positioning: I would support Tom Bacani's software development team with AI-augmented development practices, while Epi leads business automation initiatives. No overlap, no competition.

1.5 Knowledge Transfer Approach

Training methods: Onboarding workshops (Claude Code CLI setup, prompt engineering basics), pair programming sessions (hands-on problem solving on actual codebase), custom workflow design (building Agent Skills for recurring tasks), ongoing assessment and iteration.

Realistic expectations: AI agents don't replace engineering judgment—they reallocate it from implementation details to business requirements and architecture. Learning curve is weeks to proficiency, months to mastery. Some developers embrace these tools quickly; others need more time. Effective adoption requires collective buy-in, not forced adoption.

1.6 Demonstration-First Assessment

My approach: Work on real December 2025 milestone tasks using AI-first workflows, document measurable outcomes, let the team assess value based on evidence from their actual codebase—not theoretical arguments.

30-60 Day Trial Period:

- Pair programming sessions on actual Netstrata codebase challenges
- Document time savings, code quality improvements, velocity gains
- Results-based decision: if methodology accelerates milestone delivery, explore broader adoption; if fit isn't there, we have clear evidence

Why timing matters: December 2025 milestone (DMS, Tasks, Time Recording) is imminent. Early adoption provides velocity advantage when it matters most. By July 2026, team will have months of AI-augmented development experience.

Mutual assessment process:

- *Netstrata assesses*: Do AI-augmented workflows deliver measurable value? Does velocity improvement justify learning investment?
- *I assess*: Is team open to modern development practices? Will leadership support capability building?
- Better to identify alignment or misalignment through demonstration and data

Team adoption approach: Gradual and organic—start with demonstrations, let interested team members experiment, build capability progressively. Not forcing adoption, but creating environment where exploration is encouraged with leadership support. Proficiency takes weeks/months of practice.

1.7 Conclusion: Contribution Aligned with Roadmap Milestones

This AI-first methodology directly shapes how I'd contribute to Tom Bacani's team across the actual development roadmap:

1.7.1 December 2025 Release Support

"Biggest operational shift since Financials went live" - DMS, Tasks, Time Recording:

- Apply AI-first workflows to accelerate delivery
- Document measurable velocity gains for the team
- Support three squads working in parallel
- Transfer knowledge to interested team members

1.7.2 July 2026 Combined Delivery

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

- Continue AI-augmented development support
- Scale practices across squad workstreams
- Build on December 2025 learnings

1.7.3 Post-2026 Commercialization

If earlier milestone demonstrations prove value:

- Support automation and user experience enhancement
- Apply AI-first practices to external customer readiness (when needed)
- Let results from earlier milestones inform adoption decisions

The core insight: December 2025 is imminent. AI-first development offers measurable velocity gains at the most critical moment. Let production results on Netstrata's actual codebase demonstrate value.