Context Engineering for Claude Code

The INAV-Claude Project

Sensei

Making Claude Code work better through context management

The Problem

Common Problems Without Context Engineering

- ☐ Forgot to check lock files
- ☐ Used `make` instead of inav-builder agent
- ☐ Skipped testing before PR
- ☐ Didn't run code review
- ☐ Pushed to master instead of feature branch
- ☐ Missing commit message details
- ☐ Loaded 100k lines but missed the critical 100

❌ **Claude isn't bad - the process isn't structured!**

Why This Happens

```
┌───────────────────────────────────────────────┐
│ 100k Lines of Code │
│ + 5k Lines of Documentation │
│ + Build Instructions │
│ + Testing Guides │
│ + Project Tracking │
│ + Architecture Docs │
│ = 110k+ Lines Total │
└───────────────────┬───────────────────────────┘
↓
┌──────────────────────┐
│ Claude's Context │ ← Information overload
│ [saturated] │
└──────────┬───────────┘
↓
┌─────────────────┐
│ Important info │
│ gets buried! │
└─────────────────┘
```

Solution Part 1 - Role Separation

👔 MANAGER

Context Size: ~1,200 lines
Focus: Planning & Coordination
Loads: Project tracking, INDEX.md
Doesn't Load: Build instructions

📦 RELEASE MANAGER

Context Size: ~1,000 lines
Focus: Builds & Artifacts
Loads: Release workflow, changelogs
Doesn't Load: Implementation

💻 DEVELOPER

Each role sees ONLY what

Solution Part 2 - Communication System

```
# Task: Fix Terrain Data Not Loading

## Priority: HIGH

## Problem
User reports: "terrain data doesn't load" in Mission Control.

## Success Criteria                    ← Clear goals
- [ ] Root cause identified
- [ ] Terrain data loads successfully
- [ ] PR created with tests

## Available Resources                 ← What you have
- Chrome DevTools MCP available
- test-engineer agent for UI testing
```

Solution Part 3 - 12-Step Workflow + JIT Docs

```
Developer 12-Step Process: Documentation Loads:

 1. Check inbox

 2. Read task [Task file: 80 lines]

 3. Create git branch ───► [CRITICAL-BEFORE-CODE: 104 lines]
    • Check lock files

 4. Reproduce bug (fails) ───┐ • Use agents, not direct commands
    │

 5. Implement fix │ [CRITICAL-BEFORE-TEST: 113 lines]
    ├─► • Test philosophy

 6. Compile code │ • Edge cases
    │

 7. Verify fix (passes) ───┘
```

Solution Part 4 - Specialized Agents

🔨 inav-builder

Agent File: 282 lines

Represents: ~3,000 lines

Knows:

- CMake build system
- ARM cross-compilation
- Linker compatibility

Doesn't Know:

- Mission planning
- MSP protocol

🧪 test-engineer

Agent File: 492 lines

Represents: ~2,500 lines

Knows:

- Chrome DevTools Protocol
- UI testing strategies
- SITL simulator usage

Doesn't Know:

- Build systems
- Project management

How It All Fits Together

```
┌──────────────────────────────────────────────┐
│ USER: "Fix GPS bug" │
└────────────────┬─────────────────────────────┘
↓
┌────────────────────────┐
│ Developer Role │ [Focused Context]
│ Loads: 2,500 lines │
└──────────┬─────────────┘
↓
┌─────────────────────────┐
│ /start-task skill │ [Reusable Workflow]
│ • Check locks │
│ • Create branch │
└────────┬────────────────┘
↓
┌─────────────────────────────┐
│ Execute 12-Step Workflow │
└─────────┬───────────────────┘
↓
```

Hooks - Automatic Enforcement

WITHOUT HOOKS

Claude: [runs `make SITL`]

Result: ❌ Build fails

- Wrong directory
- Missing flags

User: "Use the build script"
Claude: "Oh, sorry!"

[Next task...]

Claude: [runs `make SITL` again]
User: 😠

WITH HOOKS

Claude: [tries `make SITL`]

Hook: 🛑 **INTERCEPTED**
Hook: "Use inav-builder agent"

Claude: [uses inav-builder]
Result: ✅ Build succeeds

[Next task...]

Claude: [starts typing "make"]
Hook: 🛑 **INTERCEPTED**

Real Example - Fix Terrain Data Loading

Screenshots advancing every 10 seconds (1:40 total)

0:00 → Problem: Terrain chart not displaying

0:10 → Task assigned: 80-line structured file

0:20 → Developer starts, guide loads

0:30 → test-engineer investigates with DevTools

0:40 → Root cause: plotElevation() commented out

0:50 → Fix: Chart.js v4 integration

1:00 → inav-code-review checks quality

1:10 → Success: Chart displays correctly

1:20 → PR #2518 created

1:30 → Context comparison

Results from Real-World Use

📊 PRODUCTIVITY & CONSISTENCY

Projects Completed: 78 in last 2 months

Same-Day Completions: 15+

- fix-terrain-data: 4 hours
- fix-blackbox: 1-word fix
- fix-climb-rate: 1 operator

Process Consistency: VERY HIGH

✓ Projects follow 12-step workflow

✓ Testing before PR

📉 CONTEXT EFFICIENCY

Codebase: 150,000 lines

Typical task loads:

- ~1,500 lines (with system)
- ~10-15k lines (without)

Result:

- Faster responses
- Fewer mistakes
- Better guideline adherence
- Professional dev process

5 Principles You Can Use

✓ 1. STRUCTURE BY ROLE AND PHASE

✓ 2. LOAD DOCS JUST-IN-TIME

✓ 3. USE SPECIALIZED AGENTS

✓ 4. ENFORCE WITH HOOKS

✓ 5. CLEAR COMMUNICATION BOUNDARIES

These patterns work for ANY large codebase

Self-Improvement

System Creates Its Own Tools

create-agent builds new agents!

User: "We keep looking up MSP messages"

Claude:

1. Researches MSP docs
2. Designs msp-expert agent
3. Writes agent (271 lines)
4. Updates README

Our project improves its own

Lessons Learned

Guides have self-documentation:

```
### Lessons Learned

- **Lock file format**: Include
  timestamp for debugging

- **inav-architecture first**:
  Use before Grep - saves 10min

- **SITL directory**: Use
  build_sitl/ not build/
```

System documents itself as it learns

Adapt for Your Project

🎯 THE WORKFLOW IS UNIVERSAL

12 steps work for most development projects
Firmware, web apps, data science...

The STRUCTURE is reusable
✓ Role separation
✓ JIT documentation
✓ Agent pattern
✓ Hook enforcement

The CONTENT needs customization
✗ Your build commands

📋 GETTING STARTED

Week 1: Role separation

Week 2: JIT guides

Week 3: First agent

Week 4: Add hooks

Examples:
Python/Django, React/TypeScript, Rust

Thank You!

Questions?

Repository: github.com/sensei-hacker/inav-claude
Contact: sensei-hacker on GitHub

Context engineering turns Claude from a smart assistant into a reliable, professional development team member

Backup: File Structure

```
inavflight/
├── .claude/ # Claude Code configuration
│ ├── settings.json # Hooks, permissions
│ ├── agents/ # 10 agents, 3,301 lines
│ │ ├── inav-builder.md
│ │ ├── test-engineer.md
│ │ └── msp-expert.md
│ ├── skills/ # 31 reusable workflows
│ └── hooks/ # Enforcement scripts
│
├── claude/ # Role-specific workspaces
│ ├── manager/
│ │ └── README.md # 1,200 lines of context
│ ├── developer/
│ │ ├── README.md # 2,500 lines of context
│ │ └── guides/
│ │ ├── CRITICAL-BEFORE-CODE.md (104 lines)
│ │ ├── CRITICAL-BEFORE-TEST.md (113 lines)
│ │ ├── CRITICAL-BEFORE-COMMIT.md (105 lines)
```