From Friction to Flow

Harnessing AI Assisted Development

Intro

- Dáire Treanor
- Software Engineer/Developer
- Telecoms - Airlines - Financials

Refine, refine, refine. Code

Cornerstone Documents

Iterative, interactive session with Agent to gather all of the details of the project. This process produces the Three Cornerstone documents that the project is based on.

All data is gathered from the user via CLI or dictation

Sanitise & Review

Implementation Plan Document

Iterative, interactive session with Agent to produce an implementation plan. This plan will be based on the, already produced, cornerstone documentation.

Data is gathered primarily from the cornerstone documents

Review

Phase Refinement Document

Iterative, interactive session with Agent to produce a phase implementation guide.

Data is gathered primarily from a specific phase from the Implementation Plan document but also from the user

Review

Phased Development

Request Agent to focus in on **@docs/phases/phase6.4-google-maps-integration.md** and start implementing the solution

At this point you can ask the Agent for final thoughts on the approach.

Interactive, Iterative Document Generation Process

Step 1: Build Story

Initiate Q&A session to build project story. - Drives all document generation

↓

CREATES

Step 2: Generate Cornerstone Documents

Three specialized documents based on the story:

`Architect.md (System architecture)` `Developer.md (Developer Guide)` `Tester.md (Testing Guide)`

↓

BUILDS

Step 3: Generate Implementation Plan

Synthesizes all cornerstone documents into a master plan - (Project Roadmap)

`implementation-plan.md`

↓

BROKEN DOWN INTO

Step 4: Generate Phase Guidance Files

Separate detailed guidance for each phase - (What & How)

`phase1-guidance-plan.md` `phase2-guidance-plan.md` `phase3-guidance-plan.md` `phase4-guidance-plan.md` `phase5-guidance-plan.md` ...

Subphase Development

- The subphase that you want to develop is now well defined
- Prompt Agent to implement the subphase solution
  - **Implement @docs/implementation**/phases/phase{x}/subphase/subphase-{x}.{y}
- Agent context is now very specific
- Allow Agent to proceed (Guided or YOLO)
- Run subphase test cases
- Run full test suite
- Refine as necessary

Housekeeping

- Ask Agent to bubble up to update to mark work as completed
  - @docs/implementation/phases/phase{x}/subphase/subphase-{x}.{y}
  - Verify that update has been made
- Clean up unwanted artefacts
- Run full test suite
- Commit Code
- Merge branch back down to master
- Analyse Project Structure

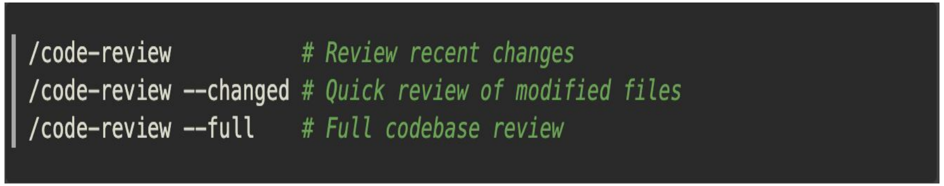
```
mcp-project-builder/
├── requirements.md
├── docs/
│   └── implementation/
│       ├── implementation-plan-v2.md
│       ├── cornerstone/
│       │   ├── architecture.md
│       │   ├── developer.md
│       │   └── tester.md
│       └── phases/
│           ├── phase0/
│           │   ├── phase0-implementation-guide.md
│           │   ├── ... (8 more .md files)
│           │   └── subphases/
│           │       └── ... (12 subphase guides)
│           ├── phase1/
│           │   ├── phase1-implementation-guide.md
│           │   ├── ... (1 more .md file)
│           │   └── subphases/
│           │       └── ... (11 subphase guides)
│           ├── phase2/
│           │   ├── phase2-implementation-guide.md
│           │   └── subphases/
│           │       └── ... (13 subphase guides)
│           ├── phase3/
│           │   ├── phase3-implementation-guide.md
│           │   ├── ... (1 more .md file)
│           │   └── subphases/
│           │       └── ... (15 subphase guides)
│           ├── ... (phase3.5, phase3.6, phase3.7, phase3.8, phase4)
│           └── phase4/
│               ├── phase4-implementation-guide.md
│               └── subphases/
│                   └── ... (10 subphase guides)
│
├── .claude/
│   ├── CLAUDE.md
│   └── settings.local.json
└── .cursor/
    └── mcp.json
```

Additional Guard Rails

- Cursor (cursor rules)
    - Project guidelines, standards, structures
- Claude
    - claude.md
        - Personal working style preferences
    - setting.local.json
        - Permissions
            - allow
            - deny
            - ask
- Living documents
    - Will change over time/duration of project
    - Local & Global

Tooling & Templating - Why?

- NextJS templating tool
  - **`npx create-nextjs-project my-awesome-project`**
  - **`npm install`**
  - **`node scripts/docs-init.js`**
  - Complete wizard
- Java templating tool
  - Use Springboot Initializer to create lean, skeleton project
  - **`npx java-microservice-template`**
  - **`npm install`**
  - **`node scripts/docs-init.js`**
  - Complete wizard
- Scripts
  - **docs-init**
  - development-strategy-init
  - quality (Linting, typescript checking, formatting)*
  - code-review
- Hooks - Pre commit checks
- Create custom / commands to run with ease
  - /docs-init
  - /dev-strategy
  - **/code-review**



```
/code-review           # Review recent changes
/code-review --changed # Quick review of modified files
/code-review --full    # Full codebase review
```

Options

Where is the Sweet spot?

- In Project CLI tooling (Non AI enabled scripts)
  - **`node scripts/docs-init.js`**

- MCP Servers (AI enabled)
  - **Code Review (run_full_review)**
    - Quality - (Linting, Formatting, Typescript checks)
    - Code Review
    - Architectural Review
    - Gap Analysis
  - **Project Builder**
    - Cornerstone document generation **(start_cornerstone_qa)**
    - Implementation plan generation **(generate-implementation-plan)**
- In Project / Commands (AI enabled)
  - ***/code-review***
  - /generate-cornerstones
  - /generate-implementation-plan
  - /generate-subphase-guide

Tips

- Docs (Cornerstones - Plan - Guides)
- Interactive sessions
- Branch for every subtask
- Merge frequently
- Create your own custom / commands
  - Code review on phase completion
  - Documentation quality (Code)
  - Documentation quality (Test)
  - Test case coverage checks
- TDD Hybrid approach
  - Write failing tests → Implement minimum code to pass → Refactor → Repeat
- Context Rot
  - /clear
  - Start new conversations
  - Minimise the number of tabs open
  - Don't compact (or whatever it's called)
- # memories
- UI - Go "Monolithic" for POC
  - FFB
  - Separation of concerns
  - UI component to generate API docs for backend team.
- Tools
  - Husky for pre commit hook checks
  - Checkstyle - Code style enforcement
  - SpotBugs - Static analysis for bugs
  - ESLint & Prettier - Code Quality
  - Turbopack - Build speed/Dev Exp
- Combine for Tooling
  - Scripts
  - / commands
  - AI
- Consider adaptive reasoning models for the reviewing processes
- Use dictation tools (Sanitize)
- MCP ???

NX - NX Cloud

Nx Cloud Self-Healing CI is an AI-powered feature that automatically detects, analyzes, and fixes CI failures without you having to babysit pull requests.

The Workflow

1. **Setup**
2. **Failure Detection**
3. **AI Analysis**
4. **Fix Generation & Verification**
5. **Review & Apply**

AI-Powered Self-Healing CI https://nx.dev/docs/features/ci-features/self-healing-ci

Juri Strumpflohner https://github.com/juristr

Thank you for your time!

daire.treanor@emsian.com