Accelerating Go Development with Claude Code: A Pragmatic Approach

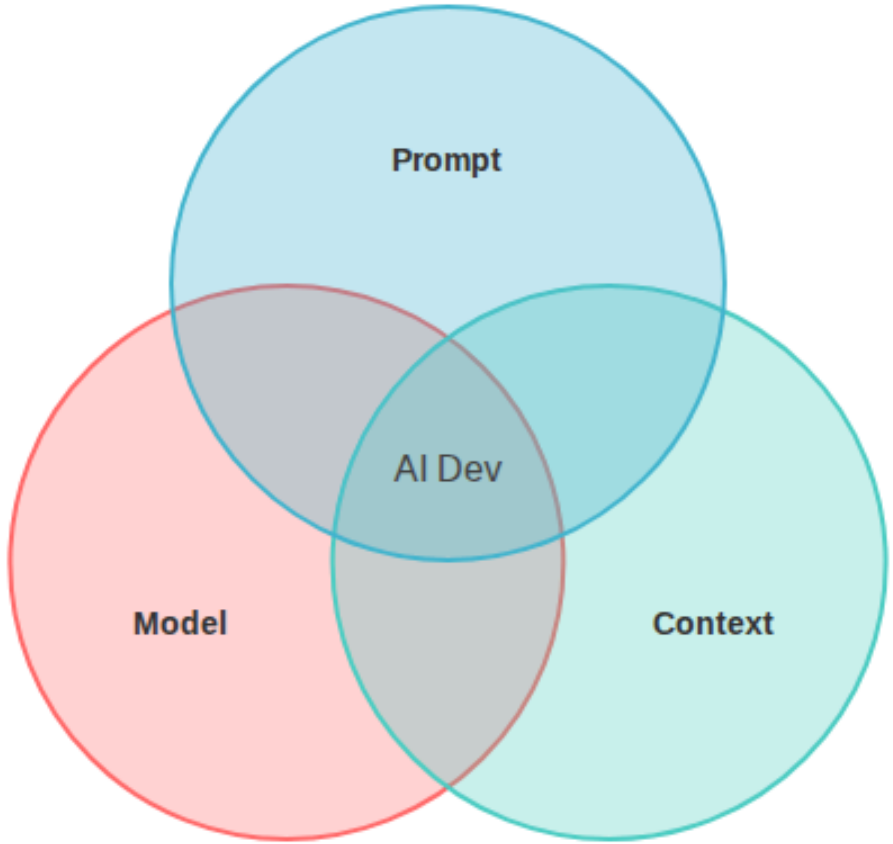
Wojciech Barczynski

Dev with AI

- Hype vs slot-machine
- Continuously evolving
- dev < dev + AI

Goal

- What works for me (so far)
- Open Discussion


Prompt
AI Dev
Model
Context

+ Tools

Models

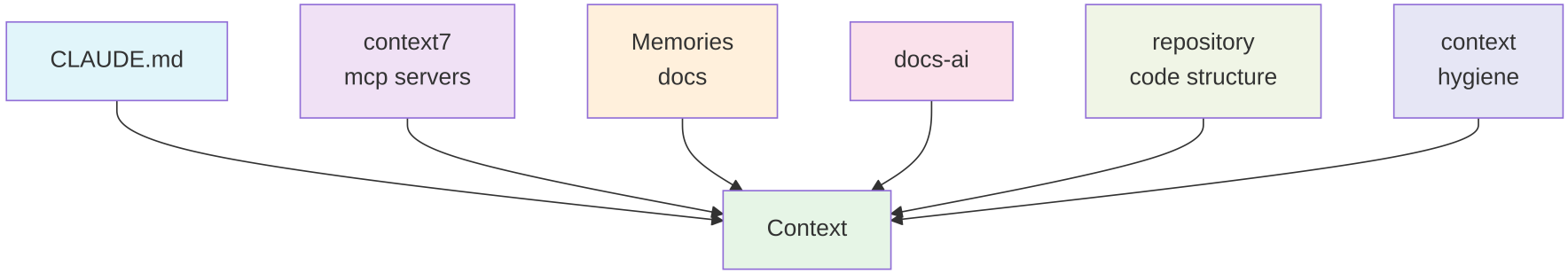
- Antrophic models lead
- `claude` → better results
- Cut-off - march 2025

Models

Models have strengths and weaknesses:

- Claude Code
- Gemini

Context


CLAUDE.md
context7
mcp servers
Memories
docs
docs-ai
repository
code structure
context
hygiene
Context

CLAUDE.md

- keep it up to date
- update after new feature*
- have a command for it

context7 mcp

- Fetches on-demand documentation and code snippets.
- Additionally:
  - Add links to prompts.
  - Add information to `memory/`.
  - Or save to `docs-ai/`.

.claude/memory

Memory (`.claude/memory`):

- Not read automatically (docs)
- Use for one-off prompts (e.g., `migration_sqlite_to_psql.md`)
- Best practices
- Keep them for later use (e.g., `memory-template`)

docs-ai / ai-docs

- More extensive docs and larger mds.
- You can link them in `CLAUDE.md`.

Repository

- Modular design
- Vertical project structure
- `CLAUDE.md` files in subfolders

Context

```
Read .claude/memory/* and ... use command ...
```

Plan.md

- Keep the model on the track
- When it double, create it
- MUST for anything more complicated
- Benefits for the model

Prompt for Claude Code

- The CLEAR Framework
- Keywords, e.g., exactly, detaile, ...
- Role-based

```
You are a [ROLE] with expertise in [DOMAIN].
Your task is to [SPECIFIC_ACTION].
```

Prompt structure Prompt 101

1. Task context
2. Tone context
3. Background data, documents, and images
4. Detailed task description & rules
5. Examples
6. Conversation history
7. Immediate task description or request
8. Thinking step by step / take a deep breath