

Accelerating Go Development with Claude Code: A Pragmatic Approach

Wojciech Barczynski (Unoperate)

Development with AI

- AI boosts productivity.
- Tools and models evolve rapidly.
- The challenge is to find what works.

Goal

- Share a pragmatic AI workflow for Go development.
- Discuss effective strategies and tools.



+ Tools

Models

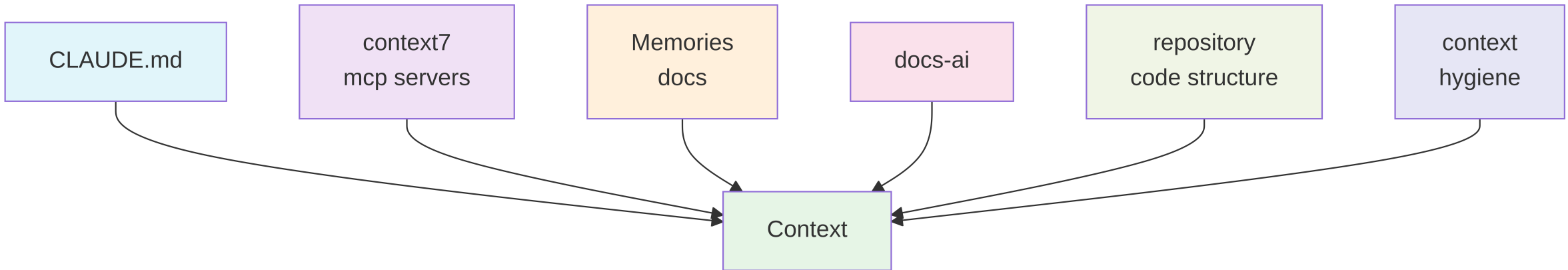
- Anthropic models lead
- `claude` → better results
- [Cut-off](#) - march 2025

Models

Models have strengths and weaknesses:

- Claude Code
- Gemini

Context



CLAUDE.md

- Keep it up-to-date.
- Update it when new features are added.
- Create a command for easy updates.

Plan.md

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

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`):

- Convention, not automatically read ([docs](#)).
- Use for one-off prompts (e.g., `migration_sqlite_to_psql.md`).
- Store best practices.
- Save prompts for future 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 ...
```

context hygiene

Once the task is complete:

- Save any essential information.
- Clear the context using the `/clear` command.

Prompt for Claude Code

- The CLEAR Framework
- Keywords, e.g., exactly, detail, [...](#)
- Role-task format pattern

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

The CLEAR Framework

- Context: Background information
- Limitations: Constraints and boundaries
- Examples: Sample inputs/outputs
- Action: Specific task to perform
- Result: Expected deliverable format

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

9. Output formatting

10. Detailed response (if any)

Will help:

- Good to watch 1-2 videos about prompt engineering
- [prompt optimizer](#) at [claude.ai](#)
- Claude can review your prompts as well.

Claude Code

- **ESC**: Provide additional information.
- **ESC ESC**: Cancel the current action.
- **Planning Mode**: Deconstruct complex tasks into smaller steps.

Tools

Claude:

- hooks
- Opentelemetry
- ccusage

Tools

Basic:

- CLI: gh
- Python - ad hoc
- mcp - few really worth it

Go-Specific AI Integration

- Go modules and dependency management
- Strict formatting (`gofmt`, `golanci-lint`)
- Testing conventions (`*_test.go`)
- Interface design patterns

Accelerating Go Development

- Continuous Effort

Accelerating Go Development

- Share the learnings with your team
- AI retrospectives

Accelerating Go Development

- More verticals in your app, the easier for the model

Accelerating Go Development

- Model
- Context
- Prompt
- Tools

Demo → Claude

github.com/wojciech12/talks & wbarczynski.pl

Thank you