Context Management Is All You Need

… when dealing with AI tools

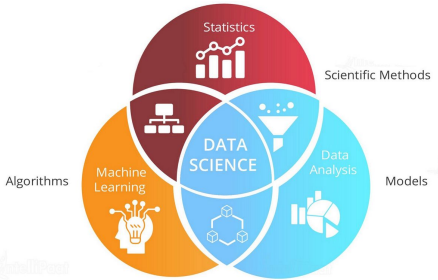
Jose Dominguez

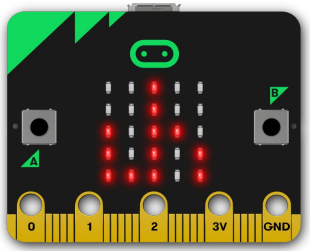
App Inventor Foundation

Product and Engineering at the App Inventor Foundation, an **Education Nonprofit** that focuses on:

- Coding through Mobile Apps
- Data Science and Physical Computing
- AI Literacy Curriculum
- Tools for teachers and Accessibility

MIT APP INVENTOR



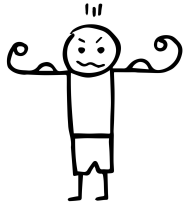


Agenda

- Back to basics
- Characteristics of LLMs
- Context and the Context Window
- AI Tools
- Vibes
- How I personally work with AI tools

The Chances

Get your raffle tickets here!



LLMs aka A Roll of the Dice

- Stochastic Parrot
- More nuanced that just predicting the next word in a sentence, but close enough
  - Interpretability: Understanding how AI models think

Why do people get mad when the LLM gets it wrong

Other characteristics of LLMs

They will always provide an answer. The answer might not make sense.

1) A simple taxonomy of LLM types

Type	What it is	Strengths	Trade-offs	Best fits
Foundational / Base	Pretrained on broad data; not aligned for following instructions.	Raw capability; best starting point for custom fine-tuning.	Poor instruction following; can be unsafe/unhelpful raw.	As a backbone for **fine-tuning**, domain adaptation, research.
Instruction-tuned (Instruct)	Base model further trained to follow natural-language instructions.	Good "do X" compliance; concise outputs.	Less conversational memory; may refuse edge cases too often.	**Batch tasks**: summarization, transformations, extraction, reporting.
Chat-aligned	Instruction-tuned + dialogue alignment and safety; often tool/function calling.	Multi-turn dialogue, clarifying questions, tools, safer defaults.	Sometimes verbose; may prioritize "helpfulness" over brevity.	**Chatbots**, assistants, live user workflows.
Reasoning / "Thinking"	Trained for deeper multi-step reasoning (often with internal scratchpad).	Strong on planning, math, coding, complex chains of steps.	Higher latency and cost; can overthink simple tasks.	**Complex planning**, agents, code generation, hard QA.
Code models	Pretrained/tuned heavily on code and tests.	Better code synthesis, repair, and test reasoning.	Weaker on open-domain chit-chat.	**Dev tools**, code refactors, SQL generation, test authoring.
Long-context models	Extended context windows (100k+ tokens).	Can ingest large docs and keep references straight.	Usually slower; context ≠ understanding; still need RAG.	**Large doc QA**, report generation, "read this corpus" tasks.
Multimodal (text+X)	Accept/emit text + images, audio, sometimes video.	Unified pipelines (OCR, chart Q&A, screenshot QA).	Tooling can be immature; cost.	**Screenshot/chat support**, slide/diagram analysis.
Structured-output / JSON-first	Tuned to emit JSON/XML reliably; sometimes with schemas.	Great for pipelines; fewer post-processing errors.	Can be brittle if schema strictness isn't guided.	**Extraction**, form-fill, workflow orchestration.
Small / Edge / Distilled	Compressed or small open-weights for on-device/server.	Low latency, low cost, data locality.	Lower raw capability; careful prompt engineering needed.	**RAG with strong retrieval**, PII-sensitive, mobile/embedded.
Mixture-of-Experts (MoE)	Router activates subsets of experts per token.	Great cost/perf scaling at large sizes.	May have load/variance quirks.	**High-throughput backends**, large assistants.


Who are
you again?

Context Window

Search Assist

A context window in large language models (LLMs) refers to the amount of text, measured in tokens, that the model can process at one time. A larger context window allows the model to consider more information, improving its ability to generate coherent and contextually relevant responses. appen.com IBM

More

Auto-generated based on listed sources. May contain inaccuracies.

Was this helpful?

Context Rot

Context Rot: https://research.trychroma.com/context-rot

NOLIMA: Long-Context Evaluation Beyond Literal Matching: https://arxiv.org/pdf/2502.05167

CHROMA TECHNICAL REPORT

July 14, 2025

Context Rot: How Increasing Input Tokens Impacts LLM Performance

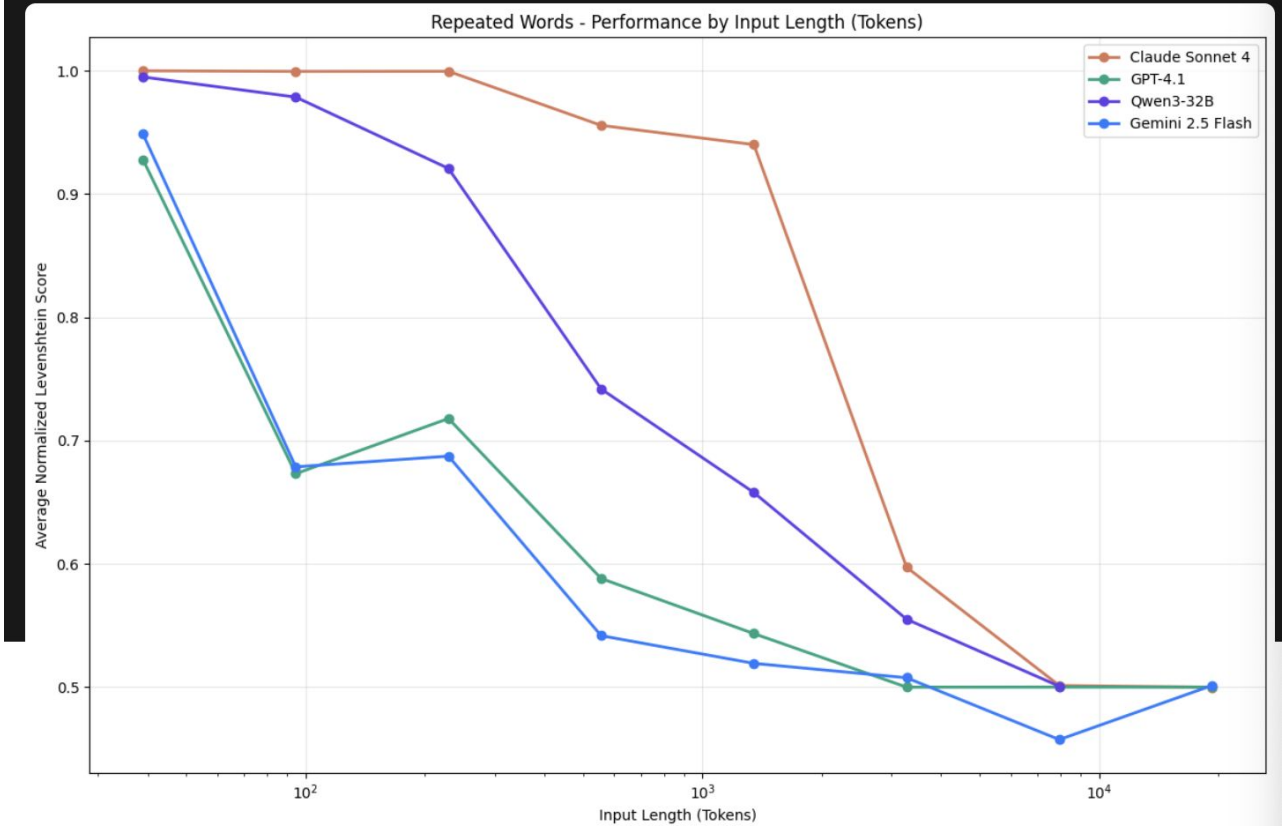
Kelly Hong Researcher - Chroma

Anton Troynikov Cofounder, Advisor - Chroma

Jeff Huber Cofounder, CEO - Chroma

Large Language Models (LLMs) are typically presumed to process context uniformly—that is, the model should handle the 10,000th token just as reliably as the 100th. However, in practice, this assumption does not hold. We observe that model performance varies significantly as input length changes, even on simple tasks.

In this report, we evaluate 18 LLMs, including the state-of-the-art GPT-4.1, Claude 4, Gemini 2.5, and Qwen3 models. Our results reveal that models do not use their context uniformly; instead, their performance grows increasingly unreliable as input length grows.



Claude Sonnet 4, GPT-4.1, Qwen3-32B, and Gemini 2.5 Flash on Repeated Words Task

Search Assist

The Average Normalized Levenshtein Similarity (ANLS) is a metric used to measure the similarity between two strings by calculating the average similarity across multiple sets of strings, taking into account the minimum number of edits needed to transform one string into another. It normalizes the Levenshtein distance to a range between 0 and 1, making it easier to interpret the similarity scores. arXiv docsaid.org

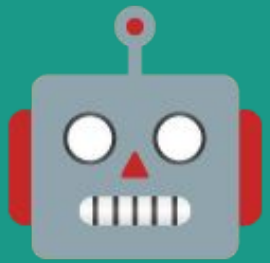
More

Models	Claimed Length	Effective Length	Base Score (×0.85: Thr.)	1K	2K	4K	8K	16K	32K
GPT-4o	128K	8K	99.3 (84.4)	98.1	98.0	95.7	89.2	81.6	69.7
Llama 3.3 70B	128K	2K	97.3 (82.7)	94.2	87.4	81.5	72.1	59.5	42.7
Llama 3.1 405B	128K	2K	94.7 (80.5)	89.0	85.0	74.5	60.1	48.4	38.0
Llama 3.1 70B	128K	2K	94.5 (80.3)	91.0	81.8	71.2	62.7	51.8	43.2
Gemini 1.5 Pro	2M	2K	92.6 (78.7)	86.4	82.7	75.4	63.9	55.5	48.2
Jamba 1.5 Mini	256K	<1K	92.4 (78.6)	76.3	74.1	70.8	62.2	52.7	43.6
Command R+	128K	<1K	90.9 (77.3)	77.0	73.5	66.2	39.5	21.3	7.4
Gemini 2.0 Flash	1M	4K	89.4 (76.0)	87.7	87.5	77.9	64.7	48.2	41.0
Mistral Large 2	128K	2K	87.9 (74.7)	86.1	85.5	73.3	51.4	32.6	18.8
Claude 3.5 Sonnet	200K	4K	87.5 (74.4)	85.4	84.0	77.6	61.7	45.7	29.8
Gemini 1.5 Flash	1M	<1K	84.7 (72.0)	68.6	61.6	51.0	44.4	35.5	28.6
GPT-4o mini	128K	<1K	84.8 (72.1)	67.7	58.2	44.2	32.6	20.6	13.7
Llama 3.1 8B	128K	1K	76.7 (65.2)	65.7	54.4	44.1	31.9	22.6	14.2

Table 3. NoLiMa benchmark results on the selected models. Following Hsieh et al. (2024), we report the effective length alongside the claimed supported context length for each model. However, we define the effective length as the maximum length at which the score remains above a threshold, set at 85% of the model's base score (shown in parentheses). Scores exceeding this threshold are underlined. Scores that are below 50% of the base score are shaded in red.

BYOM

- Way too much importance to the Model
- If I build a rig at home my code will be safe!!!
- It basically means that they found a working brain and they expect it to behave like a full human
- Haven’t really thought through how much work is involved in actually making that brain work and communicate



Terminal AI tools

Why do I prefer Claude Code, Opencode, Gemini CLI, Aider?

- Terminal based. I do everything on the terminal
- Pioneered Agentic modes (more on this later)
- It is also a departure from the idea of IDEs
- They all allow to manage context at a rather fine level

/compact
/clear

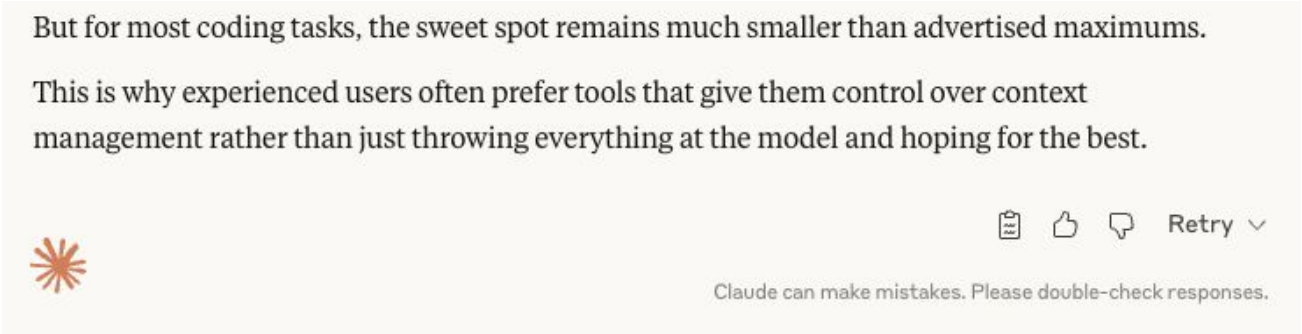
/compact

Practical Implications

This explains why tools like Claude Code include features like `compact` - even though Claude technically has a large context window, **actively managing what information is included often produces better results** than just stuffing everything in.

The research suggests that **quality and relevance of context matters far more than quantity**. A well-curated 16K token context often outperforms a sprawling 200K token context for most tasks.

/compact


But for most coding tasks, the sweet spot remains much smaller than advertised maximums.
This is why experienced users often prefer tools that give them control over context management rather than just throwing everything at the model and hoping for the best.
Retry
Claude can make mistakes. Please double-check responses.

Agentic Tools

What does *agentic* mean here

"Agentic" in this context refers to AI tools that act more like autonomous agents rather than just passive assistants. Key features:

- They can take initiative / perform multi-step workflows, not just respond to single prompts.
- They understand the broader context (project codebase, structure, goals) and can orchestrate tasks across files, repos, tools.
- They can execute or automate "routine tasks" (git operations, merging, refactoring, running tests) without you having to manually issue every little command.
- They might use tools, APIs, file manipulation, etc. So "agentic" = having agency: doing work, decision points, perhaps asking clarifying questions, using tools beyond pure text generation.

Editor Based AI tools

Cursor or Windsurf are not considered agentic (debatable, cause Cursor also has agents)

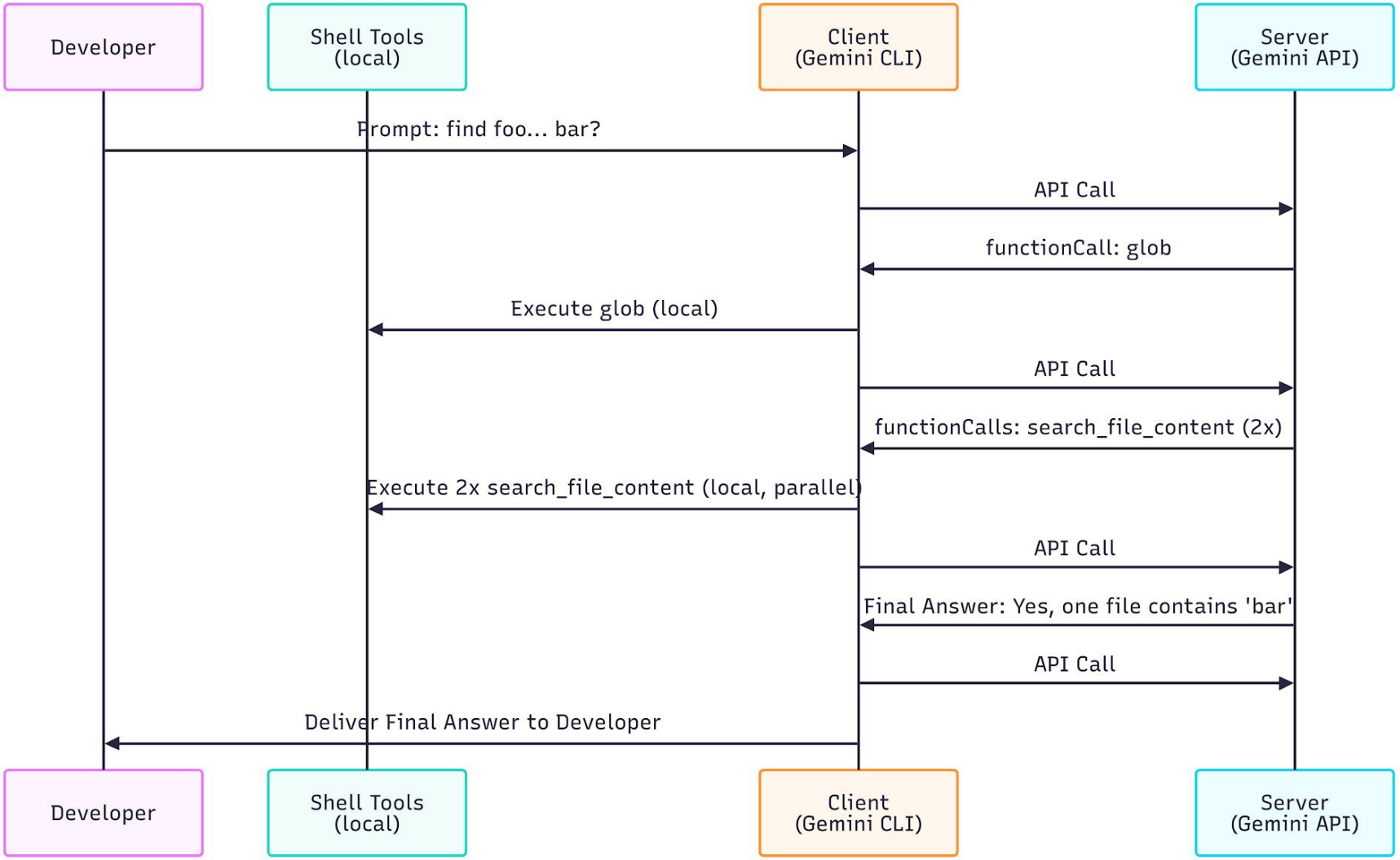
The Cline and Roo code extensions do allow for fine grain context management and the use of tools like Taskmaster(in an agentic mode)

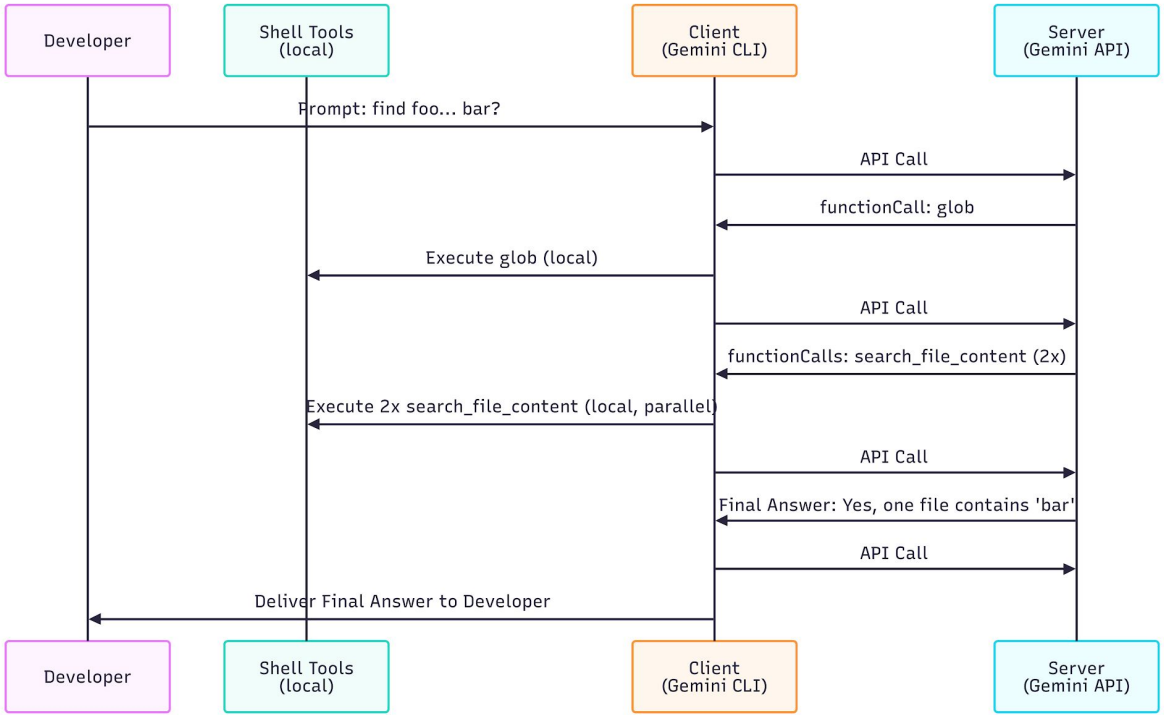
https://docs.cline.bot/getting-started/understanding-context-management

https://docs.cline.bot/features/slash-commands/smo

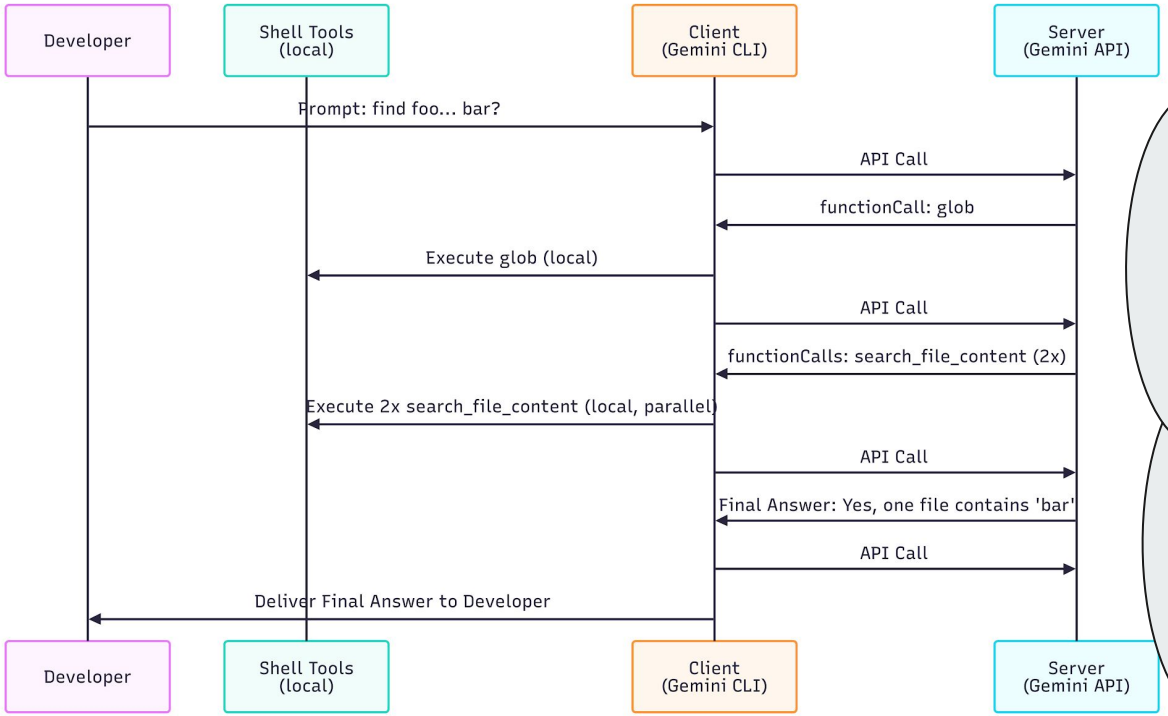
Gemini CLI Mental Model

https://softwaresecretweapons.com/opinion/gemini-cli-masterclass/gemini-cli-mental-model.html


Developer
Shell Tools (local)
Client (Gemini CLI)
Server (Gemini API)
Prompt: find foo... bar?
API Call
functionCall: glob
Execute glob (local)
API Call
functionCalls: search_file_content (2x)
Execute 2x search_file_content (local, parallel)
API Call
Final Answer: Yes, one file contains 'bar'
API Call
Deliver Final Answer to Developer
Developer
Shell Tools (local)
Client (Gemini CLI)
Server (Gemini API)



MODEL/MODELS?
AGENT/AGENTS?



A MODEL/MODELS?
AN AGENT/AGENTS?

A METAPHORICAL JOURNEY?

AN UNFOLDING ALLEGORY?

A SEARCH FOR SOMETHING LACKING (brain + heart + courage)

A HIDDEN SYSTEM WHERE A HUMAN OPERATOR IS PRETENDING TO BE A FULLY FUNCTIONAL ARTIFICIAL INTELLIGENCE?

The Vibes

Things one hears on the interwebs: Vibe Coding vs AI Assisted Coding

A number of different levels of AI; all are valid and there's some overlap:

- Zero AI: do not use AI: don't even use LSPs or any other editor tools.

Things one hears on the interwebs: Vibe Coding vs AI Assisted Coding

A number of different levels of AI; all are valid and there's some overlap:

- Zero AI: do not use AI: don't even use LSPs or any other editor tools.
- Some AI or AI assisted coding:
  - AI with either github copilot or Aider or other autocomplete tool
  - AI assisted back and forth coding: Cursor/Windsurf style
  - AI tools with Agentic behaviour (local like Claude Code or cloud based like cursor agents). Designed to work on a more autonomous way
  - Spec driven AI with tools like Amazon's Kiro or tools like Taskmaster. Designed to follow patterns in the way that engineers do

Things one hears on the interwebs: Vibe Coding vs AI Assisted Coding

A number of different levels of AI; all are valid and there's some overlap:

- Zero AI: do not use AI: don't even use LSPs or any other editor tools.
- Some AI or AI assisted coding:
  - AI with either github copilot or Aider or other autocomplete tool
  - AI assisted back and forth coding: Cursor/Windsurf style
  - AI tools with Agentic behaviour (local like Claude Code or cloud based like cursor agents). Designed to work on a more autonomous way
  - Spec driven AI with tools like Amazon's Kiro or tools like Taskmaster. Designed to follow patterns in the way that engineers do
- All AI:
  - Unsupervised AI coding == Vibe till you die…

Things one hears on the interwebs: Vibe Coding vs AI Assisted Coding

A number of different levels of AI; all are valid and there's some overlap:

- Zero AI: do not use AI: don't even use LSPs or any other editor tools.
- Some AI or AI assisted coding:
  - AI with either github copilot or Aider or other autocomplete tool
  - AI assisted back and forth coding: Cursor/Windsurf style
  - AI tools with Agentic behaviour (local like Claude Code or cloud based like cursor agents). Designed to work on a more autonomous way
  - Spec driven AI with tools like Amazon's Kiro or tools like Taskmaster. Designed to follow patterns in the way that engineers do
- All AI:
  - Unsupervised AI coding == Vibe till you die… at times literally… when you roll the dice and your agent deletes the production database and actually tells you what it's done!

My Own Personal Vibes

Types of projects (or how I think about them)

3 different axis:

- How much do you know about the underlying technology?
- How much do you know about the business domain?
- How much do you care about the results? [Longer term maintainability]

Types of projects (or how I think about them)

3 different axis:

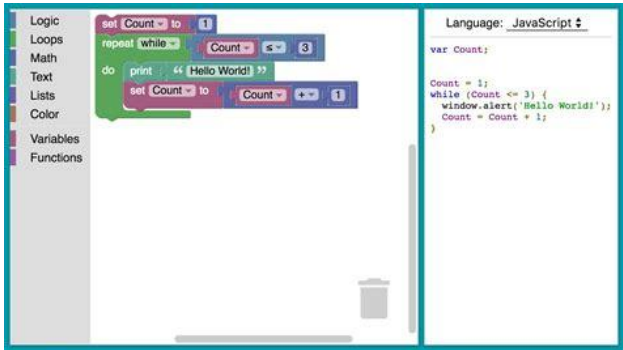
- How much do you know about the underlying technology?
  - How much do you know about the business domain?
  - How much do you care about the results?
- How much do you know about the business domain?
  - How much do you know about the underlying technology?
  - How much do you care about the results?
- How much you care about the results?
  - How much do you know about the underlying technology?
  - How much do you know about the business domain?

Some of my [pet] projects

A Blockly plugin for RTC (Google docs style)

Fully vibed project:

- How much do you know about the business domain?
  - Quite a bit. I would know how it is supposed to behave and how to test this
- How much do you know about the underlying technology?
  - Know JS/TS well, but I am not so familiar with libraries like Y.js for CRDT
- How much do you care about the results?
  - Not terribly; it is a prototype after all



Micro-Sentree

Fully vibed project:

- How much do you know about the underlying technology?
  - I would know how to build this myself, given the time
    - Svelte + SvelteKit + drizzle + postgresql
    - Plain JS library to integrate in the monitored app
- How much do you know about the business domain?
  - Quite a bit. I would know how it is supposed to behave and how to test this
- How much do you care about the results?
  - Not terribly; it is a prototype after all

Cassan Earth: Agentic Earth Observation Platform

Half vibed project:

- How much do you know about the underlying technology?
  - Front end: I would know how to build this myself, given the time
    - The Stochastic Parrot made me choose NextJS here. Fully (supervised) vibed
  - Back end: It is a bunch of agents written with Google's ADK, which we had never used before. Fully written by hand
- How much do you know about the business domain?
  - Very little when it comes to actual earth observation
- How much do you care about the results?
  - Not terribly; it is a prototype after all

Some Learnings

When building products, coding is rarely the bottleneck. This held up in the past, but holds now even more.

Blockly plugin: main constraint is the number of people you can comfortably have in one session sharing one document until it all falls apart

Micro-Sentree: If we ever need to scale the application being monitored, we'd also need to scale the monitoring system, which again is not a coding task

Cassan Earth: If I don't know how things work underneath, I cannot evaluate what the AI tool is producing

Some Learnings

When building products, coding is rarely the bottleneck. This held up in the past, but holds now even more.

Micro-Sentree: If we ever need to scale the application being monitored, we'd also need to scale the monitoring system, which again is not a coding task

Cassan Earth: If I don't know how things work underneath, I cannot evaluate what the AI tool is producing

Supervising the Chances and the Vibes

Luck is what happens when Preparation meets Opportunity

Likely not Seneca

How do I work with Claude code?

I prefer to “supervise” Claude; why and how?

- Interrupt often
  - Learn to read Claude’s output Matrix style
- Manage context; this is the most important aspect as we have already discussed
  - Try to avoid auto-compact and /compact
- Always track progress
  - Use a spec driven pattern for features: This doesn’t have to be a full blown system like Kiro or Taskmaster because I do not want to be the bottleneck.

Track progress

Always track progress, even for small features:

- Claude has a “planning mode” which will only “think” and not code.
- Ask Claude in planning mode to think through the feature. You can use words like “Think hard” or “Overthink”
- Ask Claude to come up with a spec that is:
    - Technical in nature
    - Detailed enough so Claude can work autonomously at a different stage
    - Use MCP like Context7 or fetch to look at documentation
    - It is divided into phases or stages, and tasks, so Claude can keep track of what has been and what needs to be done
    - Ask Claude to add all your development guidelines to the spec

Track progress

Always track progress, even for small features:

- Claude has a “planning mode” which will only “think” and not code.
- Ask Claude in planning mode to think through the feature. You can use words like “Think hard” or “Overthink”
- Ask Claude to come up with a spec that is:
    - Technical in nature
    - Detailed enough so Claude can work autonomously at a different stage
    - Use MCP like Context7 or fetch to look at documentation
    - It is divided into phases or stages, and tasks, so Claude can keep track of what has been and what needs to be done
    - Ask Claude to add all your development guidelines to the spec
- Ask Claude to write the spec down before any other work starts
- Review the spec and make manual edits or ask Claude to make edits
- Close the session and start a new one with a brand new context

Track progress

Always track progress, even for small features:

- Claude has a “planning mode” which will only “think” and not code.
- Ask Claude in planning mode to think through the feature. You can use words like “Think hard” or “Overthink”
- Ask Claude to come up with a spec that is:
    - Technical in nature
    - Detailed enough so Claude can work autonomously at a different stage
    - Use MCP like Context7 or fetch to look at documentation
    - It is divided into phases or stages, and tasks, so Claude can keep track of what has been and what needs to be done
    - Ask Claude to add all your development guidelines to the spec
- Ask Claude to write the spec down before any other work starts
- Review the spec and make manual edits or ask Claude to make edits
- Close the session and start a new one with a brand new context
- Ask Claude to read the spec and start working on it stage by stage, and document back in the spec the progress at the end of each stage.
- Within each state read Claude’s output as if it were the Matrix
- After each stage make sure the output matches what was supposed to be done

Inspiration exists, but it has to find you working

Likely Pablo Picasso

Use your time wisely

If it was as easy as getting the AI tools to do the work for you, anyone could do it and you wouldn’t relevant anymore

The landscape of what many of us do has already changed

Coding is not the bottleneck

Train yourself to do other things that bring value

Other tricks

The model/tool will behave slightly different on each release

The tool will lie to you

Use MCP but watch out for the context window being blown up

Use git worktrees if you want to, but for me, I become the bottleneck

Subagents in Claude Code; I don’t particularly use them

Tricky to do live Evals, but are they useful

Want some more?

TDD, AI Agents and coding with Kent Beck

"I've changed my mind on AI coding" – Adam Wathan (creator of Tailwind)

If you are into benchmarks and the LLM flavour of the month watch Gosu Coder

If you are into agents: Karpathy vs. McKinsey: The Truth About AI Agents (Software 3.0) by Nate B Jones

Questions?