

AI Engineering Workflow

HUANG Jingyi

Sino-french Engineer School
Beihang University

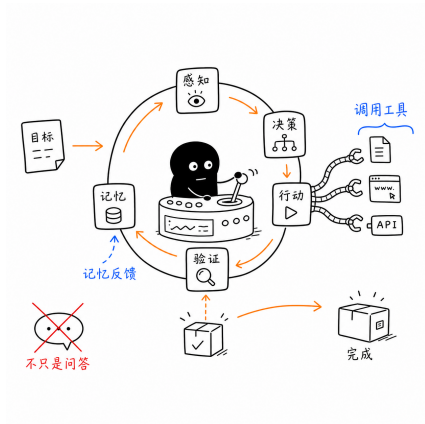
June 24, 2026

Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase

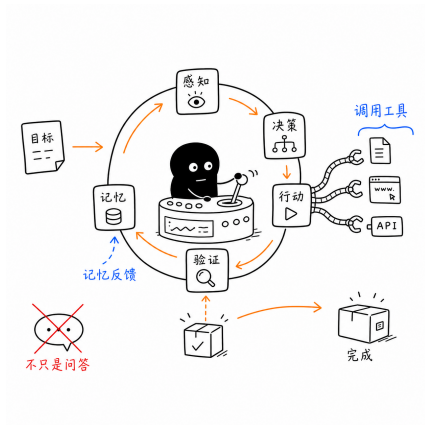
What Is an AI Agent?

- An **AI agent** = LLM + tools + environment + autonomy.



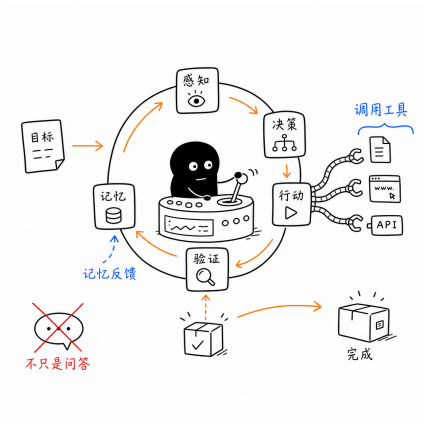
What Is an AI Agent?

- An **AI agent** = LLM + tools + environment + autonomy.
- Unlike a chatbot, it can **act**: read files, edit code, run tests, commit.



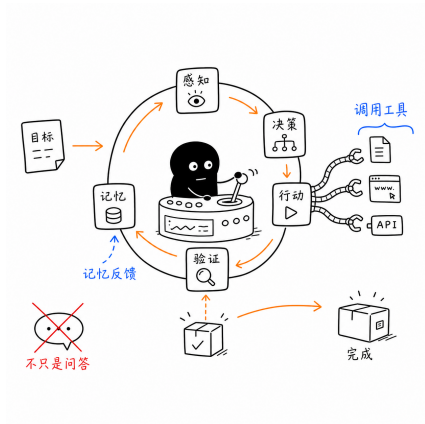
What Is an AI Agent?

- An **AI agent** = LLM + tools + environment + autonomy.
- Unlike a chatbot, it can **act**: read files, edit code, run tests, commit.
- A **coding agent** lives inside your codebase and changes it.



What Is an AI Agent?

- An **AI agent** = LLM + tools + environment + autonomy.
- Unlike a chatbot, it can **act**: read files, edit code, run tests, commit.
- A **coding agent** lives inside your codebase and changes it.
- Your job shifts from typing every line to **directing the agent**.



Examples of AI Agents

Coding Agents: live in your codebase



Claude Code

Terminal agent by Anthropic



OpenAI Codex

IDE and CLI coding agent



Kimi Code

Moonshot coding assistant

General Agents: work across apps and channels



OpenClaw

Personal AI assistant on your devices



Hermes Agent

Self-improving agent by Nous Research

What Is a Skill?

- A **skill** is a reusable prompt or playbook for an agent.



What Is a Skill?

- A **skill** is a reusable prompt or playbook for an agent.
- It tells the agent **how** to perform a specific task.



What Is a Skill?

- A **skill** is a reusable prompt or playbook for an agent.
- It tells the agent **how** to perform a specific task.
- Benefits: save context tokens, enforce consistency, share team knowledge.



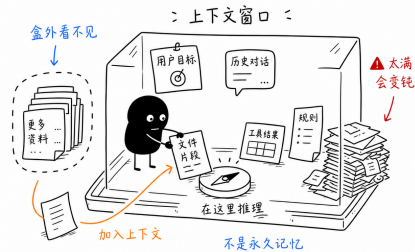
What Is a Skill?

- A **skill** is a reusable prompt or playbook for an agent.
- It tells the agent **how** to perform a specific task.
- Benefits: save context tokens, enforce consistency, share team knowledge.
- In Claude Code / Codex: invoke with `/skill-name`.



Context Window: The Agent's Workspace

- The **context window** is everything the LLM can “see” right now.
- It includes:
 - ▶ **System prompt**: always present, keep it tiny
 - ▶ **Conversation history**: questions, answers, code
- A typical coding session has phases:
 1. **Explore**: read the codebase
 2. **Implement**: write code
 3. **Test**: run feedback loops
- As the window fills up, the LLM enters the **dumb zone**.



AI Agents: Engineering Teammates

- We all feel AI is changing everything.



AI Agents: Engineering Teammates

- We all feel AI is changing everything.
- But the fundamentals of software engineering still apply.



AI Agents: Engineering Teammates

- We all feel AI is changing everything.
- **But the fundamentals of software engineering still apply.**
- An AI agent is not a replacement for a developer.



AI Agents: Engineering Teammates

- We all feel AI is changing everything.
- **But the fundamentals of software engineering still apply.**
- An AI agent is not a replacement for a developer.
- It is a teammate that amplifies good engineering.



AI Agents: Engineering Teammates

- We all feel AI is changing everything.
- **But the fundamentals of software engineering still apply.**
- An AI agent is not a replacement for a developer.
- It is a teammate that amplifies good engineering.
- The same principles that make human teams work also make AI teams work.

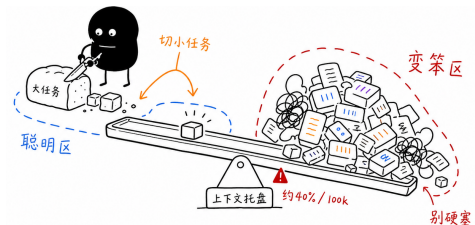


Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs**
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase

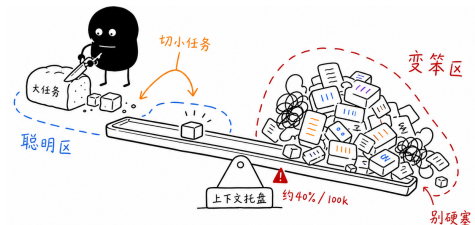
Constraints of LLMs

- LLMs have a **smart zone** and a **dumb zone**.



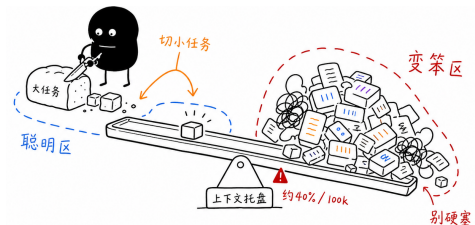
Constraints of LLMs

- LLMs have a **smart zone** and a **dumb zone**.
- LLMs continually forget.



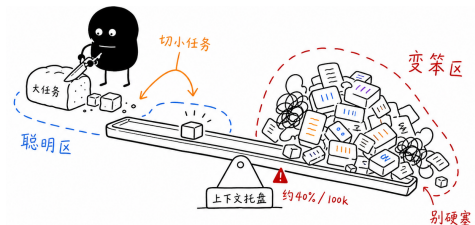
Constraints of LLMs

- LLMs have a **smart zone** and a **dumb zone**.
- LLMs continually forget.
- Attention scales quadratically with tokens.



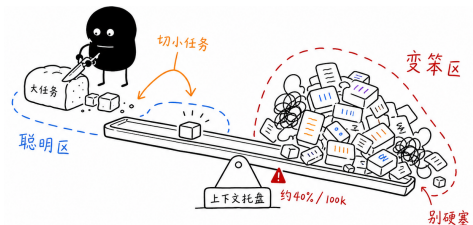
Constraints of LLMs

- LLMs have a **smart zone** and a **dumb zone**.
- LLMs continually forget.
- Attention scales quadratically with tokens.
- Around **100K tokens**, quality starts to degrade.



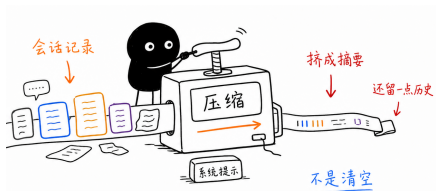
Constraints of LLMs

- LLMs have a **smart zone** and a **dumb zone**.
- LLMs continually forget.
- Attention scales quadratically with tokens.
- Around **100K tokens**, quality starts to degrade.
- **Implication:** keep tasks small enough to fit inside the smart zone.



Clearing vs Compacting

Compacting

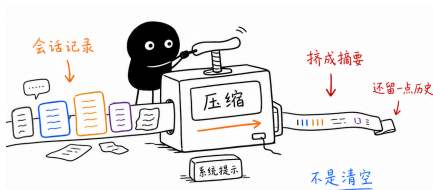


Squeezes the conversation into a summary.

Lossy. Full detail is gone.

Clearing vs Compacting

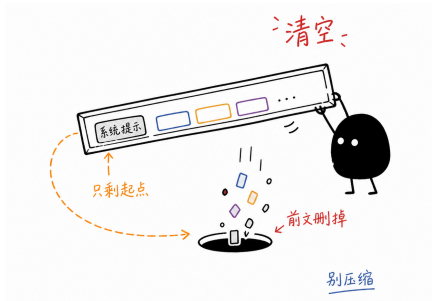
Compacting



Squeezes the conversation into a summary.

Lossy. Full detail is gone.

Clearing

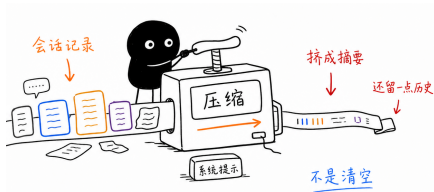


Throws away conversation and starts fresh.

Clean. Reproducible. Same state.

Clearing vs Compacting

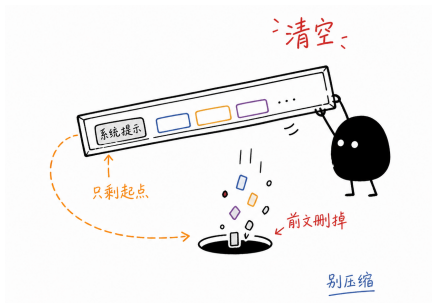
Compacting



Squeezes the conversation into a summary.

Lossy. Full detail is gone.

Clearing



Throws away conversation and starts fresh.

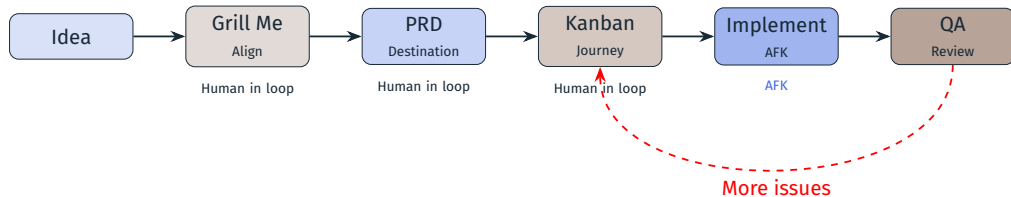
Clean. Reproducible. Same state.

Prefer clearing: it gives a deterministic reset point.

Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow**
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase

From Idea to Implementation



Day shift (human plans) → **Night shift** (AI implements)

Two Types of Tasks

Human-in-the-Loop Tasks

- Understanding the idea
- Grilling for alignment
- Reviewing the PRD
- Shaping the Kanban board
- Manual QA and taste

These need judgment, taste, and shared understanding.

Two Types of Tasks

Human-in-the-Loop Tasks

- Understanding the idea
- Grilling for alignment
- Reviewing the PRD
- Shaping the Kanban board
- Manual QA and taste

These need judgment, taste, and shared understanding.

AFK Tasks

- Exploring the codebase
- Writing migrations
- Implementing features
- Running tests and type checks
- Generating commits

These can run unattended once the plan is clear.

Two Types of Tasks

Human-in-the-Loop Tasks

- Understanding the idea
- Grilling for alignment
- Reviewing the PRD
- Shaping the Kanban board
- Manual QA and taste

These need judgment, taste, and shared understanding.

AFK Tasks

- Exploring the codebase
- Writing migrations
- Implementing features
- Running tests and type checks
- Generating commits

These can run unattended once the plan is clear.

The goal is to queue up as much work as possible for the night shift.

Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment**
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase

Why “Specs-to-Code” Fails

Specs-to-Code

1. Write a spec document
2. Paste it into an LLM
3. Get code back
4. If the code is wrong, fix the spec
5. Repeat

You are ignoring the code.

Why it fails: You never look at the implementation, so you cannot tell if the AI misunderstood. The code drifts away from reality.

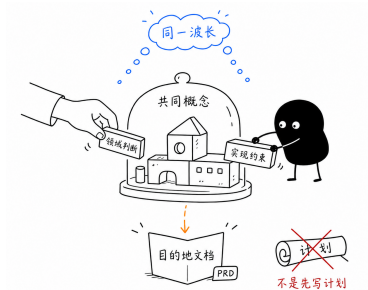
Why “Specs-to-Code” Fails

Specs-to-Code

1. Write a spec document
2. Paste it into an LLM
3. Get code back
4. If the code is wrong, fix the spec
5. Repeat

You are ignoring the code.

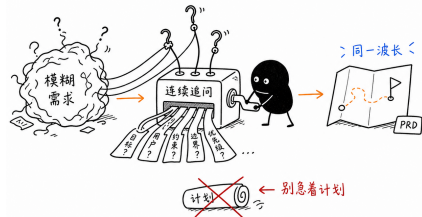
Why it fails: You never look at the implementation, so you cannot tell if the AI misunderstood. The code drifts away from reality.



Shared design concept beats a one-way plan.

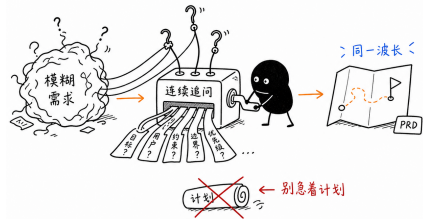
Grill Me: Reach a Shared Design Concept

- With AI, you need a **shared understanding**, not just a plan.



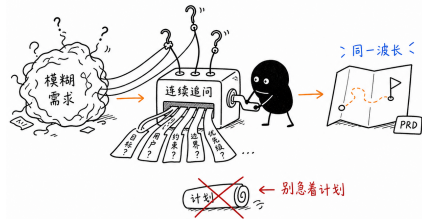
Grill Me: Reach a Shared Design Concept

- With AI, you need a **shared understanding**, not just a plan.
- Grill Me interviews you relentlessly, one question at a time.



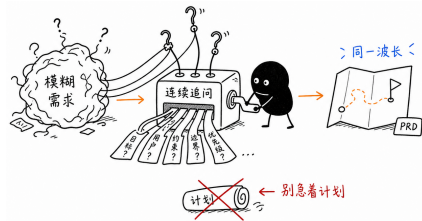
Grill Me: Reach a Shared Design Concept

- With AI, you need a **shared understanding**, not just a plan.
- Grill Me interviews you relentlessly, one question at a time.
- The conversation history becomes the design concept asset.



Grill Me: Reach a Shared Design Concept

- With AI, you need a **shared understanding**, not just a plan.
- Grill Me interviews you relentlessly, one question at a time.
- The conversation history becomes the design concept asset.



You are not buying a plan. You are getting on the same wavelength as AI.

Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban**
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase

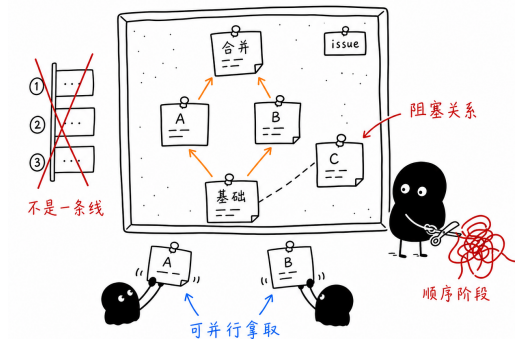
Two Documents: Destination and Journey

PRD = Destination Document

- Problem statement
- Solution overview
- User stories
- Implementation decisions
- Testing decisions
- Things **out of scope**

Summarizes the shared design concept.

Kanban = Journey Document



Break work into small, independent issues with clear blocking links.

Beware of Doc Rot: old PRDs in the repo mislead future AI runs.

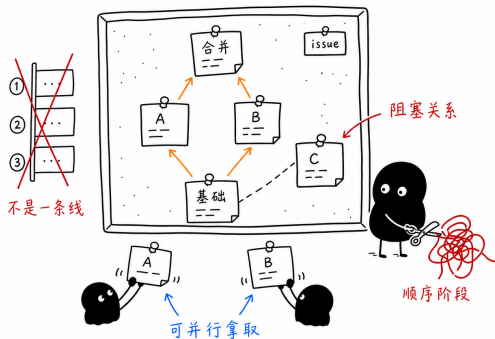
Two Documents: Destination and Journey

PRD = Destination Document

- Problem statement
- Solution overview
- User stories
- Implementation decisions
- Testing decisions
- Things **out of scope**

Summarizes the shared design concept.

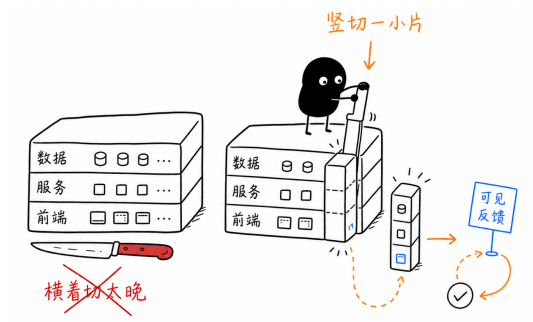
Kanban = Journey Document



Break work into small, independent issues with clear blocking links.

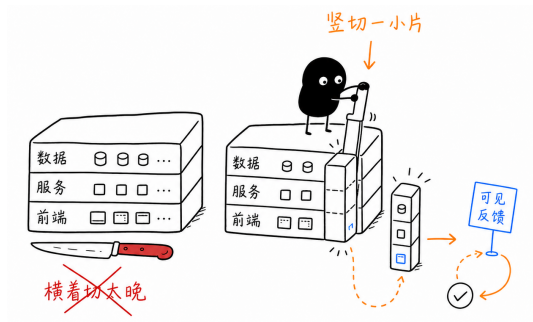
Beware of Doc Rot: old PRDs in the repo mislead future AI runs.

Vertical Slices: Tracer Bullets



AI loves to code horizontally. You must force it vertical.

Vertical Slices: Tracer Bullets

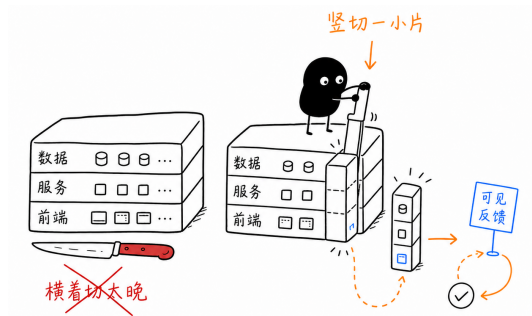


Horizontal Slices (Bad)

Build one full layer at a time.
No feedback until Phase 3.
AI codes blind.

AI loves to code horizontally. You must force it vertical.

Vertical Slices: Tracer Bullets



Horizontal Slices (Bad)

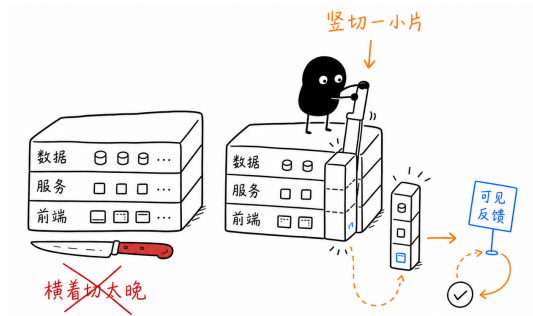
Build one full layer at a time.
No feedback until Phase 3.
AI codes blind.

Vertical Slices (Good)

Instant feedback on every layer.
Like tracer bullets in the night sky.

AI loves to code horizontally. You must force it vertical.

Vertical Slices: Tracer Bullets



Horizontal Slices (Bad)

Build one full layer at a time.
No feedback until Phase 3.
AI codes blind.

Vertical Slices (Good)

Instant feedback on every layer.
Like tracer bullets in the night sky.

AI loves to code horizontally. You must force it vertical.

Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution**
- 7 Phase 4: Own the Codebase

The Ralph Loop

- Once the board is ready, the human can step back.



Loop until backlog is clear

The Ralph Loop

- Once the board is ready, the human can step back.
- **Issue-driven:** tasks come from local Markdown issue files.



Loop until backlog is clear

The Ralph Loop

- Once the board is ready, the human can step back.
- **Issue-driven:** tasks come from local Markdown issue files.
- **Autonomous selection:** agent picks the next task by status and dependencies.



Loop until backlog is clear

The Ralph Loop

- Once the board is ready, the human can step back.
- **Issue-driven:** tasks come from local Markdown issue files.
- **Autonomous selection:** agent picks the next task by status and dependencies.
- **Implement-feedback-fix:** code, test, iterate until green.



Loop until backlog is clear

The Ralph Loop

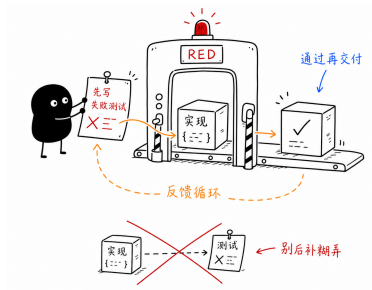
- Once the board is ready, the human can step back.
- **Issue-driven:** tasks come from local Markdown issue files.
- **Autonomous selection:** agent picks the next task by status and dependencies.
- **Implement–feedback–fix:** code, test, iterate until green.
- **Stop signal:** finish when AFK tasks are done.



Loop until backlog is clear

TDD Is Essential for AI

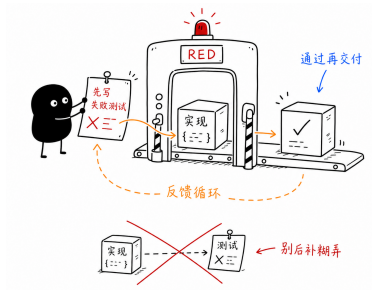
- Without tests, AI codes totally blind.



Red-Green-Refactor cycle

TDD Is Essential for AI

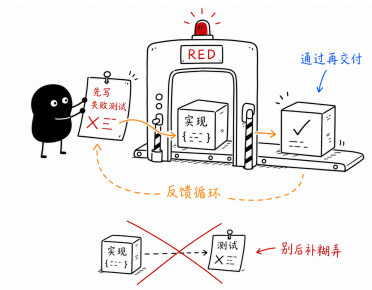
- Without tests, AI codes totally blind.
- The quality of your feedback loops is the ceiling of AI output.



Red-Green-Refactor cycle

TDD Is Essential for AI

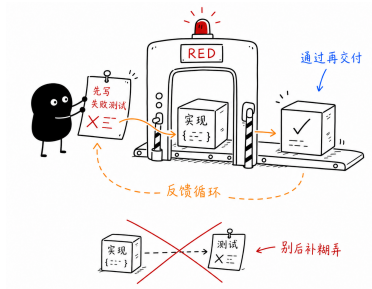
- Without tests, AI codes totally blind.
- The quality of your feedback loops is the ceiling of AI output.
- TDD: Red (failing test) → Green (passing code) → Refactor.



Red-Green-Refactor cycle

TDD Is Essential for AI

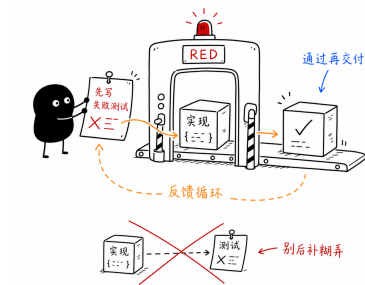
- Without tests, AI codes totally blind.
- The quality of your feedback loops is the ceiling of AI output.
- TDD: Red (failing test) → Green (passing code) → Refactor.
- Writing the test first makes it harder for AI to cheat.



Red-Green-Refactor cycle

TDD Is Essential for AI

- Without tests, AI codes totally blind.
- The quality of your feedback loops is the ceiling of AI output.
- TDD: Red (failing test) → Green (passing code) → Refactor.
- Writing the test first makes it harder for AI to cheat.



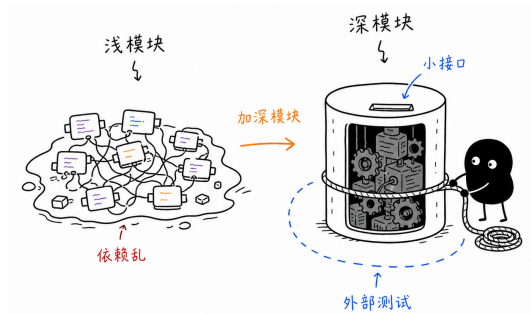
Red-Green-Refactor cycle

If your codebase has no feedback loops, you will never get decent AI output.

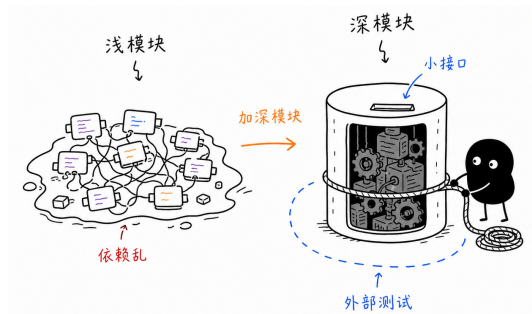
Outline

- 1 A Different Way to Think About AI
- 2 Constraints of LLMs
- 3 The Full Workflow
- 4 Phase 1: Grill for Alignment
- 5 Phase 2: PRD and Kanban
- 6 Phase 3: AFK Execution
- 7 Phase 4: Own the Codebase**

Deep vs Shallow Modules



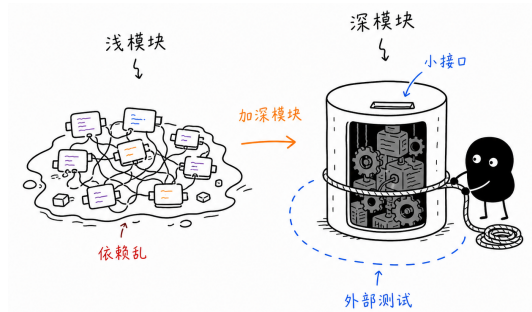
Deep vs Shallow Modules



Shallow Modules

Many small files, tangled deps.
Hard for AI to navigate.

Deep vs Shallow Modules



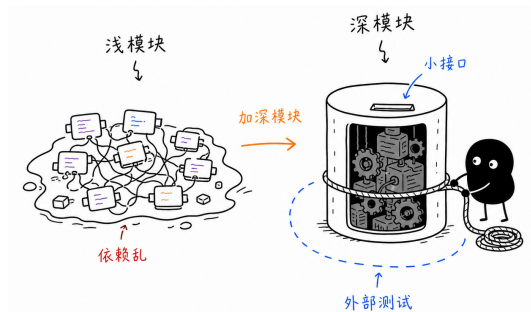
Shallow Modules

Many small files, tangled deps.
Hard for AI to navigate.

Deep Modules

Small interface, rich internals.
Easy to test.

Deep vs Shallow Modules



Shallow Modules

Many small files, tangled deps.
Hard for AI to navigate.

Deep Modules

Small interface, rich internals.
Easy to test.

Unaided, AI produces shallow modules. You must direct it toward deep ones.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.

Retain your sense of the codebase while preserving your sanity.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.
- The antidote: keep a **module map** in your head.

Retain your sense of the codebase while preserving your sanity.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.
- The antidote: keep a **module map** in your head.
- Know the shape of each module—what it does and how it behaves.

Retain your sense of the codebase while preserving your sanity.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.
- The antidote: keep a **module map** in your head.
- Know the shape of each module—what it does and how it behaves.
- **Design the interface yourself.**

Retain your sense of the codebase while preserving your sanity.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.
- The antidote: keep a **module map** in your head.
- Know the shape of each module—what it does and how it behaves.
- **Design the interface yourself.**
- **Delegate the implementation to AI.**

Retain your sense of the codebase while preserving your sanity.

Design Interfaces, Delegate Implementation

- Many developers feel they know their codebase less well since adopting AI.
- The antidote: keep a **module map** in your head.
- Know the shape of each module—what it does and how it behaves.
- **Design the interface yourself.**
- **Delegate the implementation to AI.**
- Modules become “gray boxes”: known shape, unknown internals.

Retain your sense of the codebase while preserving your sanity.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.
- Now you QA manually.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.
- Now you QA manually.
- QA is how you impose your taste back onto the code.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.
- Now you QA manually.
- QA is how you impose your taste back onto the code.
- If you try to automate everything—idea generation, planning, QA—you end up with slop.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.
- Now you QA manually.
- **QA is how you impose your taste back onto the code.**
- If you try to automate everything—idea generation, planning, QA—you end up with slop.
- Human taste is the difference between working software and software that feels right.

QA Is Where Taste Enters

- After the AFK loop finishes, you get a commit summary.
- Now you QA manually.
- **QA is how you impose your taste back onto the code.**
- If you try to automate everything—idea generation, planning, QA—you end up with slop.
- Human taste is the difference between working software and software that feels right.

We are not producing slop. We are producing high-quality work.

Thank You!

Questions & Discussion

Content: Matt Pocock, aihero.dev

Illustrations: [ian-xiaohei-illustrations](#) skill, Codex