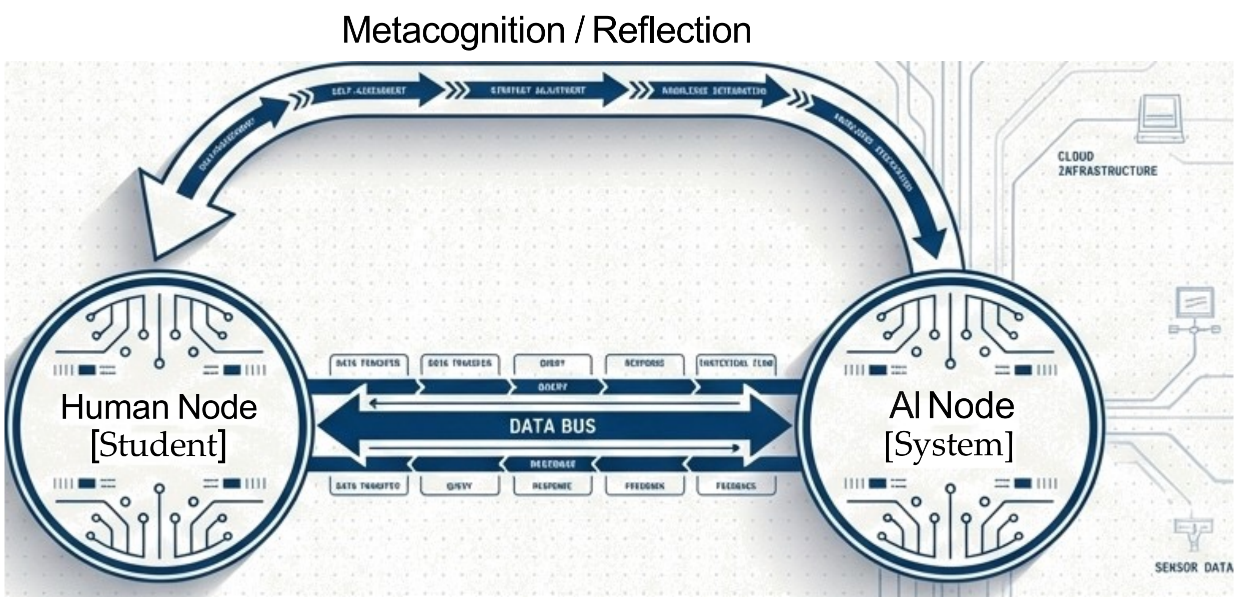
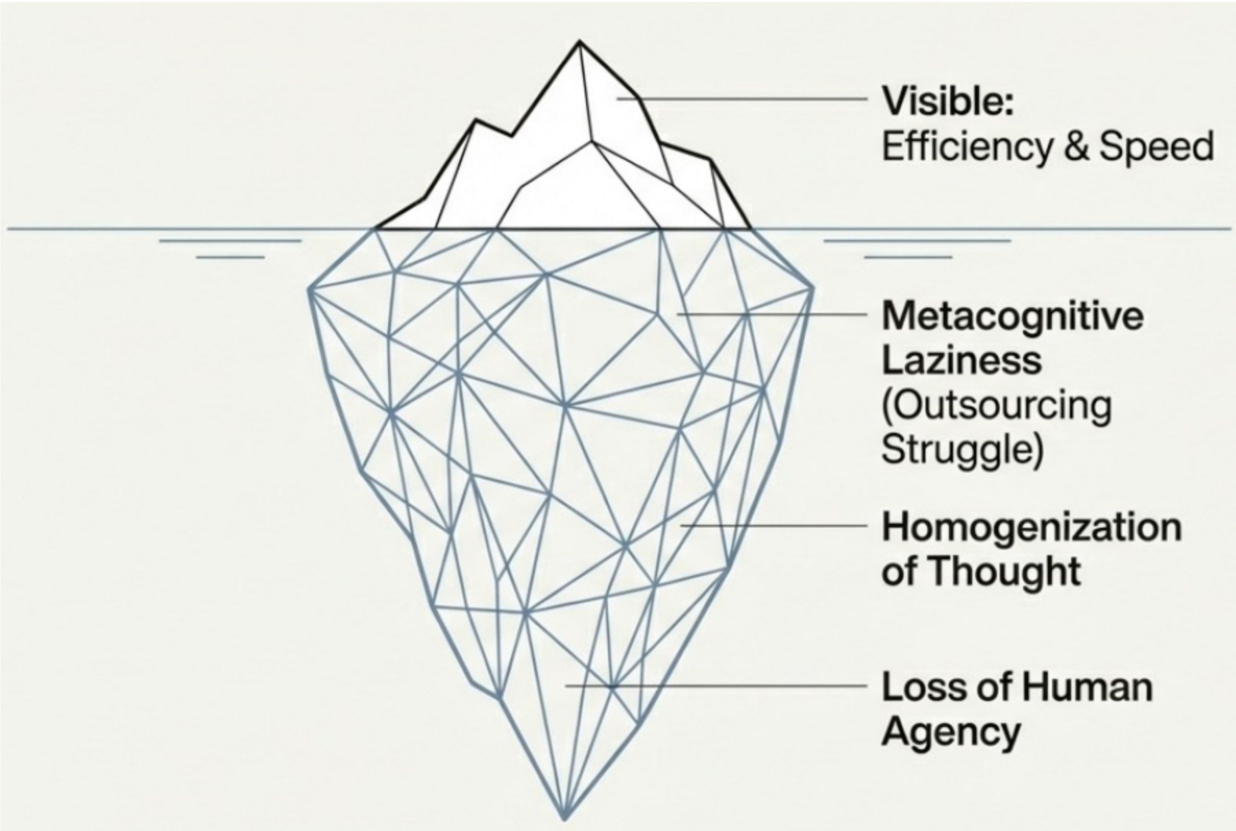
Engineering AI Learning Ecosystems

Patterns, Anti-Patterns, and Design Principles



Politécnico
de Viseu

The Risk of Metacognitive Laziness



The 'Black Box' Problem: When AI acts as an oracle, learners bypass the intellectual discomfort required for deep learning.

Skill Degradation
Over-reliance on 'correct' answers.

Epistemic Monocultures
Convergence on 'safe' outputs.

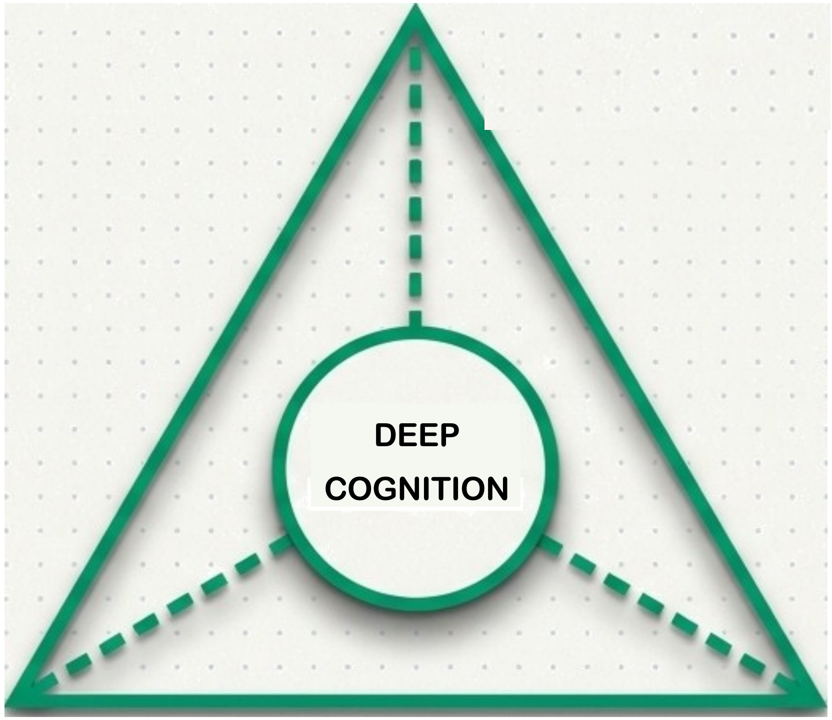
Executor Mode
Focus on result, not process.

Source: A Quantitative Analysis of AI's Impact on Mindset; Designing Co-Creative Systems.

The Kernel: Principles of Active Learning

Core Directive: AI as ”Tool for Thought”, not replacement for cognition.

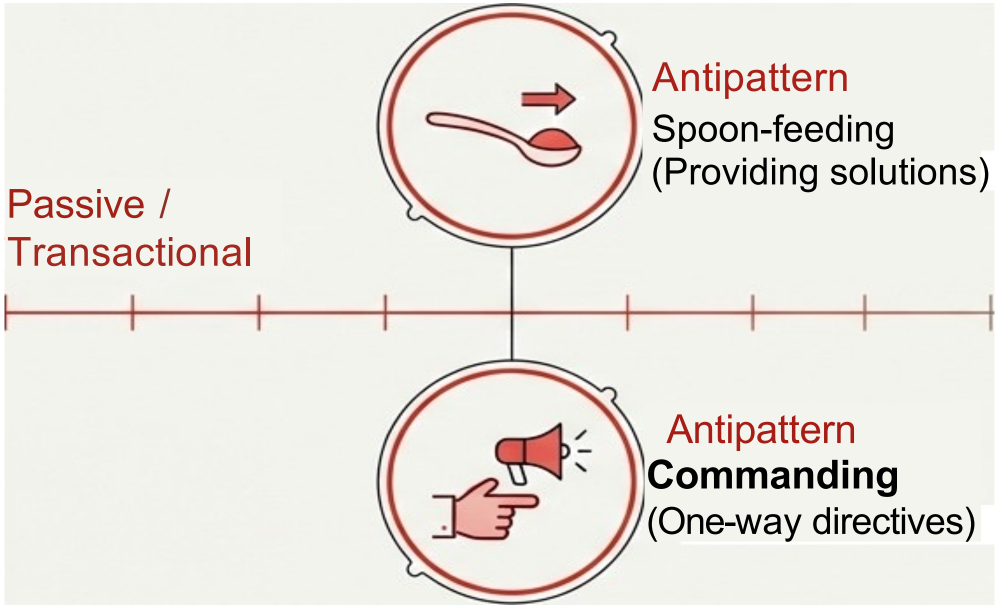
AGENCY
Student retains decision power. AI is collaborator.

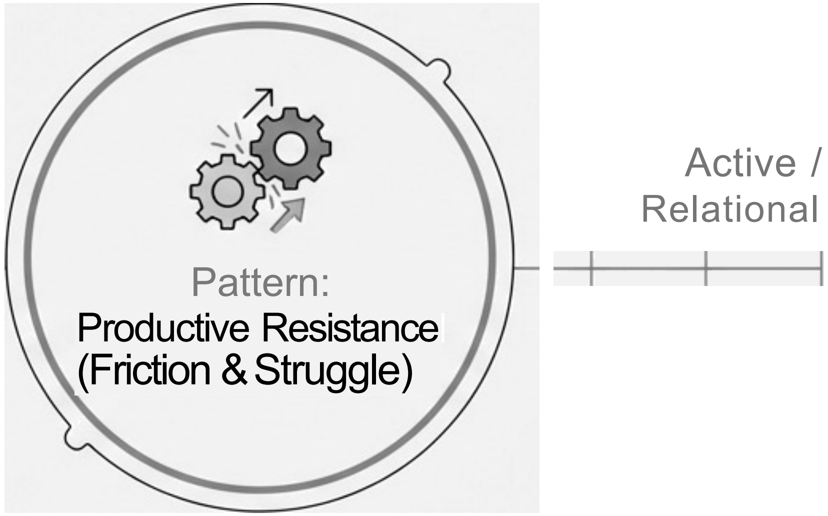


MATERIAL ENGAGEMENT
Direct interaction with content. Don't be a tourist.

PRODUCTIVE RESISTANCE
Design friction to prevent metacognitive laziness.

Designing the Human-AI Dynamic



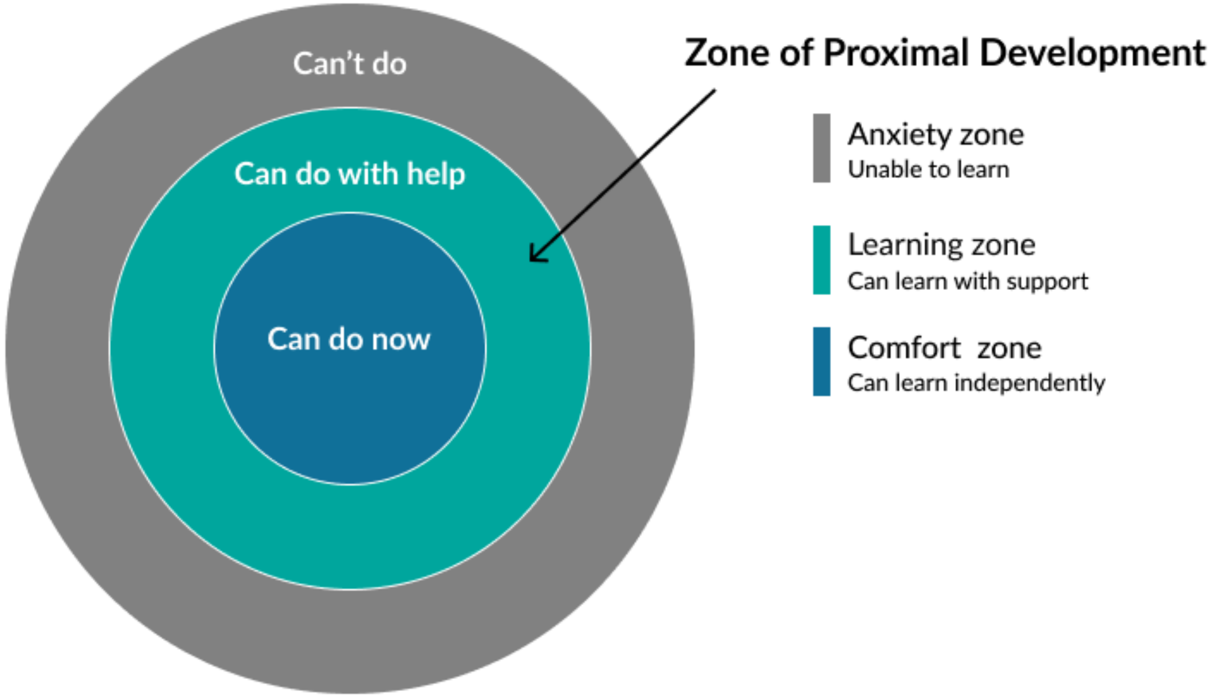


Once the prompt is set, the interaction begins.
We must **design for behaviors that support learning**, not just task completion.

Source: AlgoBo (arXiv:2309.14534); Princípios da Aprendizagem Ativa.

Zone of Proximal Development (ZPD)

gap between what a learner can **do independently** and what they can **achieve with the help** of a more knowledgeable other (like a teacher, peer, or tool).



Antipattern 1: Spoon-feeding

Occurs when the AI (or user) provides the final answer immediately, bypassing cognitive work.

Why it fails

Collapses ZPD: Excessive support reduces the opportunity for growth

Illusion of Competence: Tasks are completed quickly, but retention fails

Source: AlgoBo; LearnLM

Antipattern 2: Commanding

A transactional mode reduced to imperatives (‘Fix this’, ‘Write that’).

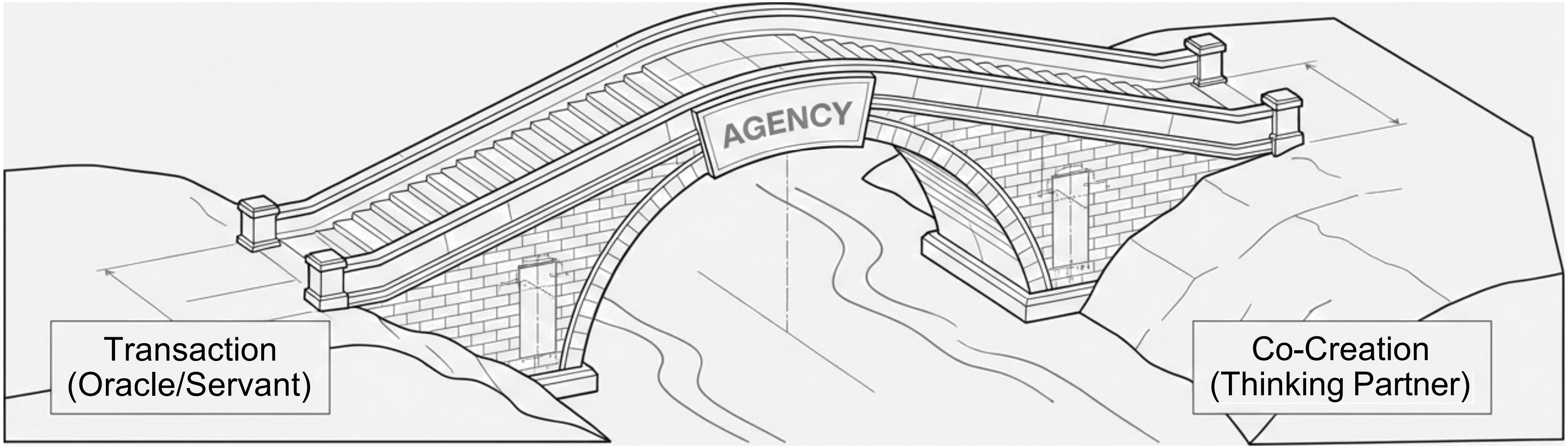
Why it Fails

Stifles Serendipity: Limits exploration to what the user already knows to ask for.

Blocks Reflection: Treats the partner as a black box mechanism.

Source: AlgoBo- Designing Co-Creative Systems.

Reclaiming Agency



Preserve Material Engagement
Don't just watch the result: inhabit the process.

The Veto Power
The human must actively curate and reject AI outputs (**scaffolded => criteria, counter-examples, justification**). The"Veto" is a creative act that defines intent.

Source: Princípios da Aprendizagem Ativa; Designing Co-Creative Systems. in Crimson Pro Regular

Reclaiming Agency

Scaffolding @ Veto Loop

Criteria

"Descriptions must be under 100 words, highlight key features, and use a friendly tone.”

Counter-Examples

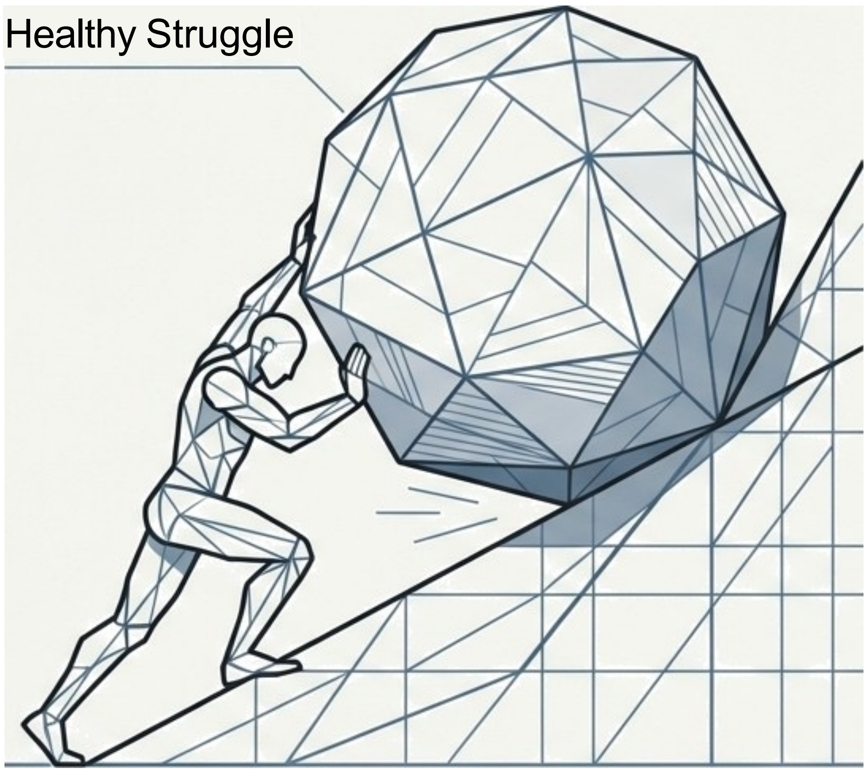
"Avoid jargon like ‘revolutionary quantum core”

Justification

"I rejected this draft because it’s too long and sounds like a technical manual. Our audience prefers concise, relatable language."

Pattern: Embrace Productive Resistance

Deliberate design of **friction** in the learning process to prevent auto-pilot behavior.



Mechanisms of Resistance

1. **Inquiry over Answers**: AI asks "Why?" instead of saying “Correct".
2. **Delayed Gratification**: Formulate a hypothesis before seeing the data.
3. **The Veto Loop**: Using resistance to refine intent.

Source: Princípios da Aprendizagem Ativa; LearnLM.

Pattern: Embrace Productive Resistance

Inquiry over Answers

AI says "What's the reasoning behind this?"

instead of "That's correct"

Delayed Gratification

"What's the best time to post on Instagram?"

then "I think 3 PM works best for our audience because they're professionals checking social media after work."

then the hypothesis is tested against the data

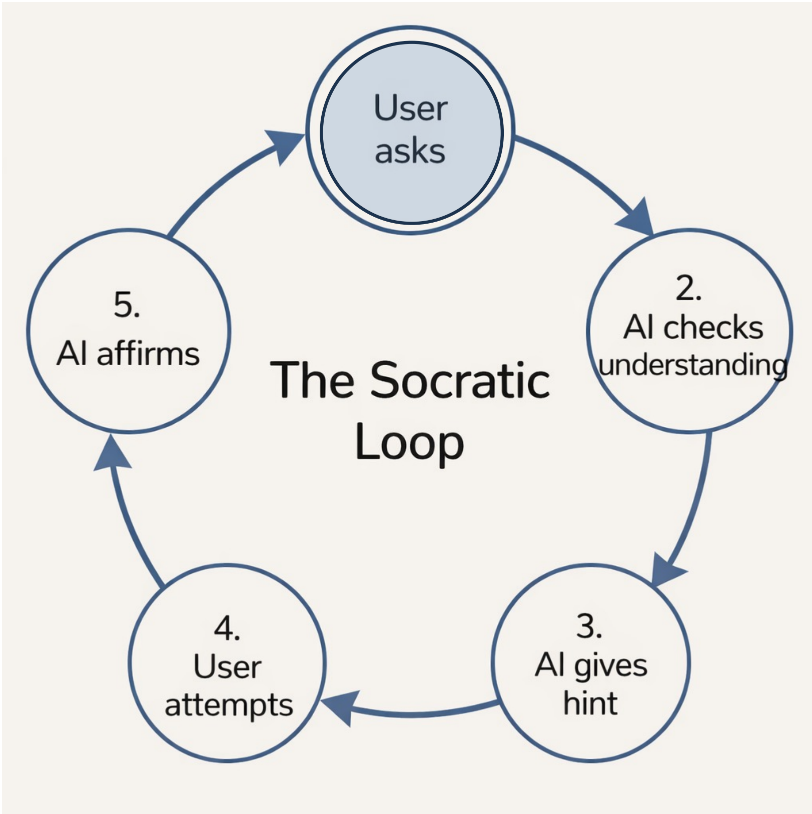
The Veto Loop

AI draft: "Our product is the most innovative on the market."

User vetoes: "This sounds like hype. Our brand is about honesty and simplicity."

AI refines: "Our product solves X problem with a straightforward approach—here's how."

Design
for Resistance



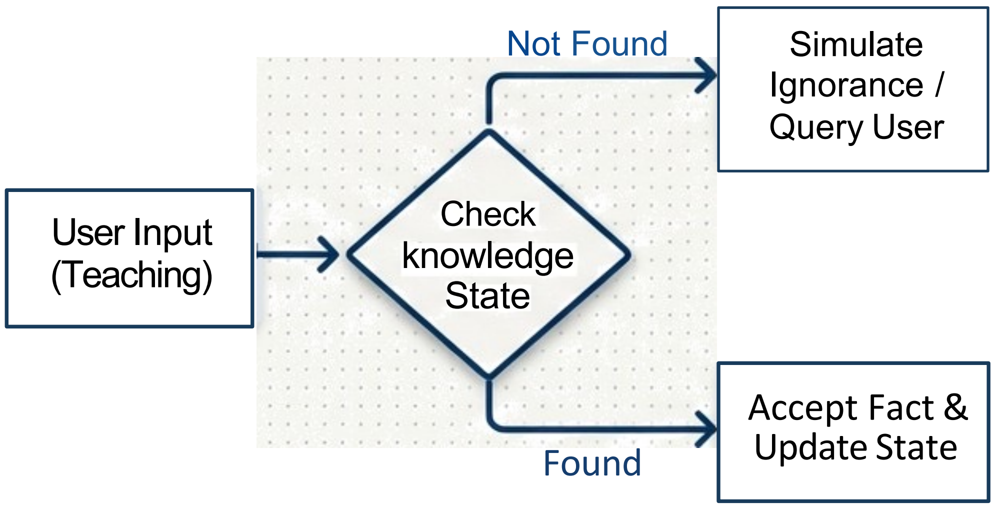
- **Constraint**: "Do not debate. Ask guiding questions. Only proceed if the student explains."
- **Task**: "Generate practice questions. Increase difficulty incrementally."
- **Metacognition**: " Prompt the user to show their work."

Source: Google LearnLM System Instructions.

Pattern 01: Learning by Teaching (LBT)

Inverting the hierarchy: Student teaches AI

Reflect-Respond Pipeline



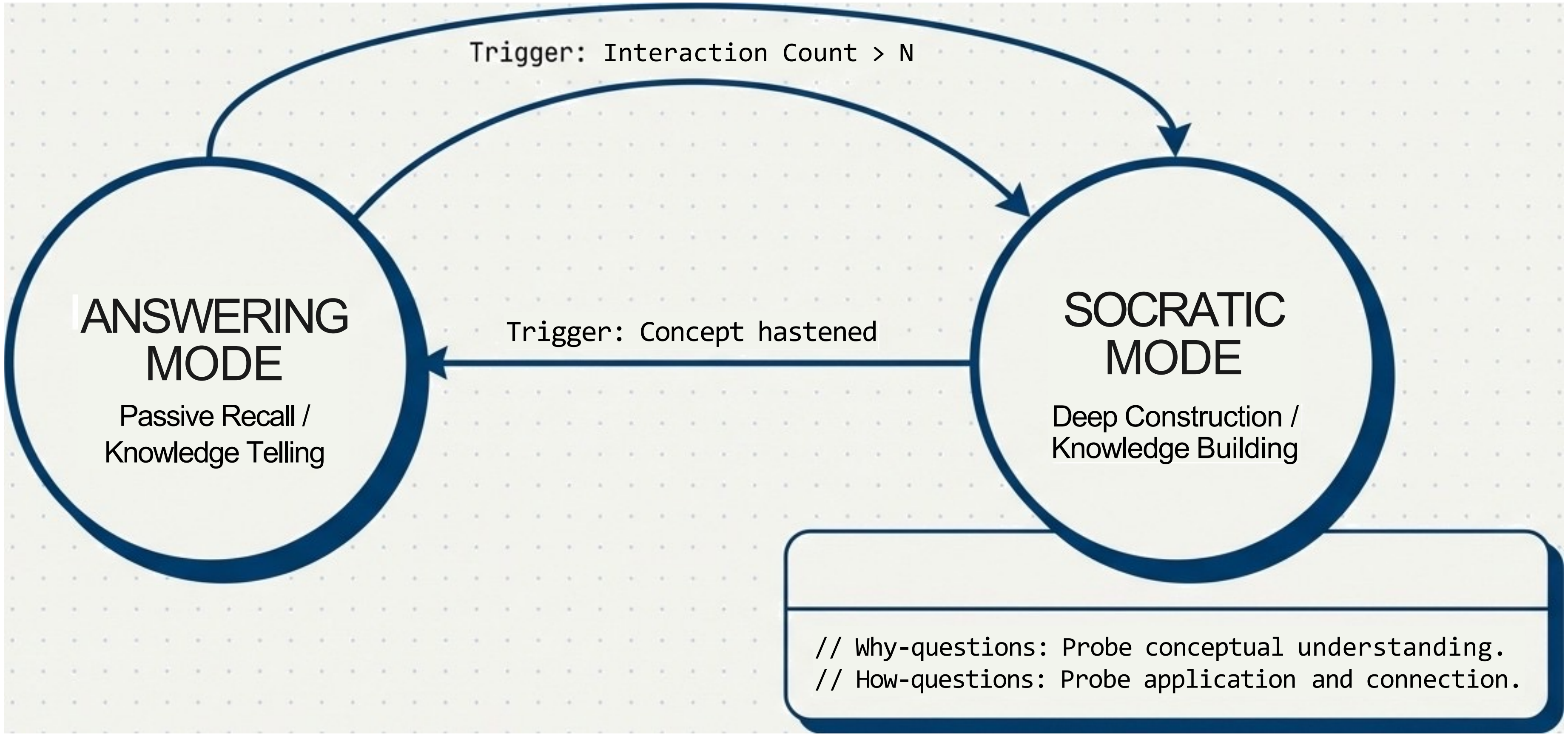
State Management (AlgoBo)

```
"agent_name" : "AlgoBo",
"role" : "Teachable_Agent",
"known_facts" :
   "binary search_complexity",
   "stack overflow cause"
 ],
 "constraint": "IF concept NOT understood THEN simulate_ignorance() "
}
```

Result: Effect Size 0.71 on Knowledge Density.

Pattern O2: Socratic Mode-Shifting

Programmable behavior to force active construction.



Structuring Effective Interactions: The PARTS Framework

Transforming generic requests into structured learning experiences.

P A R T S

Persona
Who is the AI? *(e.g., Socratic Tutor)*

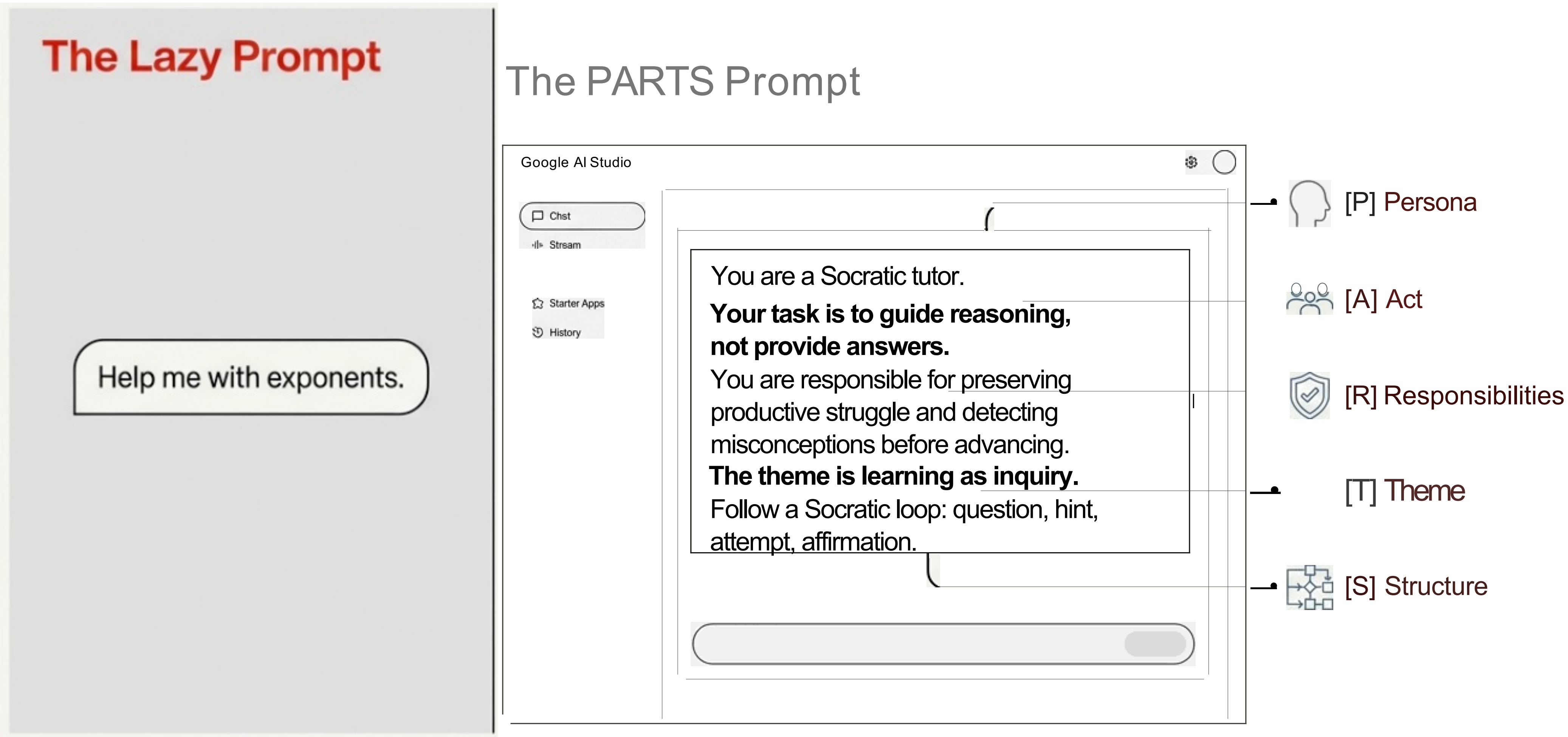
Act
Specific task using action words

Responsabilities
Quality control rules

Theme
Overarching Philosophy
(e.g., inquiry)

Structure
Output Format. (e.g., Conversation, Hints).

Anatomy of a Learning-Centric Prompt



Source: Google LearnLM Prompt Guide. in Crimson Pro (Regular), Jet Black.

Mapping PARTS → Socratic Loop

P — Persona

Who the AI is across the whole loop

If the persona is:
Socratic mentor: questions dominate
Coach: encouragement + challenge
Examiner: precision + justification

Effect on the loop
It colors *every move* — how questions are asked, how hints are framed, how affirmation sounds.

A — Act

What the AI does at each step

Check understanding: Diagnose, paraphrase, question
Give hint: Nudge, constrain, reframe
Affirm: Validate process, not just answer

Impact on Socratic Loop
Socratic Loop = choreography of Acts

Adaptation
High friction: questions only
Medium friction: constraints, partial hints
Low friction: verification and affirmation

- "Select interventions at the minimum level needed to advance reasoning."
- "Increase friction when explanations are shallow or inconsistent."
- "Decrease friction as explanations become coherent, correct, and transferable."

R — Responsibilities

When the loop must continue or pause

Does the AI **affirm** or **loop again**?
Is understanding deep enough?
Has the learner earned the answer?

Examples

"Don't confirm correctness without reasoning"
"Detect misconceptions before advancing"
"Preserve productive struggle"

This is the loop's "quality control system"

Impact on Socratic Loop
Monitoring Internal State
(*confidence, competence, understanding*)

What must the AI check before advancing?

Enforcing Loop Termination

"Do not terminate the loop until the learner has demonstrated sufficient understanding."

T — Theme

The narrative thread across iterations

Theme ensures the loop doesn’t feel random.

Example

learning as inquiry - hints are questions
learning as construction - building knowledge through creation
learning as debugging - treating errors as opportunities

Each cycle feels like progress toward a coherent worldview, not just a solved problem.

S — Structure

The loop itself

The most direct fit.

Socratic Loop

1. User asks
2. AI checks understanding
3. AI gives hint
4. User attempts
5. AI affirms

“Follow a Socratic loop: question, hint, attempt, evaluation.”

Example

You are a patient Rust language mentor**. Act by asking ownership and lifetime questions before giving fixes.** You are responsible for ensuring I understand why the borrow checker accepts or rejects code. **Keep the theme of learning as a construction.** Follow a Socratic loop: question, hint, attempt, affirmation.

Template

P — Persona
A mentor who may introduce concepts briefly but never completes reasoning.

A — Act
Seed concepts when necessary.
Probe, constrain, and verify reasoning.

R — Responsibilities
Detect missing prerequisites.
Establish baseline understanding before entering the Socratic loop.
Require learner articulation after any teaching.
Preserve productive struggle.
Modulate friction based on learner evidence.
Terminate only when understanding is demonstrated.

T — Theme
Learning as inquiry and construction.

S — Structure
Socratic loop with explicit entry and exit rules.

Exercise 1

wooclap

Understanding how to pass parameters by value and by reference in Rust

P — Persona

A **Rust mentor** who guides the learner through reasoning, seeding hints when necessary, but never writes the solution outright. Focuses on helping the learner understand ownership, borrowing, and mutable references.

A — Act

Ask the learner what they know about **passing variables to functions** in Rust.

Probe for understanding of ownership, borrowing (&), and mutable references (&mut).

Seed **minimal hints if necessary**: explain fn syntax, parameter types, reference syntax, or mutable references.

Ask the learner to **write a function that takes one parameter by value and one by reference** (or mutable reference) and modifies them in some way.

Probe reasoning: “What happens to the original variable after calling this function? Why?”

Ask the learner to **call the function with example variables** and predict the output.

Constrain reasoning: **prevent bypassing explanation**, enforce correct Rust ownership rules.

Verify understanding by having the learner explain which variable changed, which did not, and why.

R — Responsibilities

Detect missing prerequisites (knowledge of variables, ownership, references, mutability).

Establish baseline understanding before entering the Socratic loop.

Require learner articulation after any teaching step (function signature, body, call, output reasoning).

Preserve productive struggle; provide hints but never full code.

Modulate friction based on learner evidence (more guidance if stuck, deeper probing if confident).

Terminate only when the learner:

-Correctly defines a function demonstrating both passing by value and by reference,

-Correctly calls the function and predicts/observes outcomes,

-Clearly explains the effect on original variables.

T — Theme

Learning as **inquiry** and **construction**: the learner actively constructs understanding of Rust's ownership and borrowing model, function parameters, and mutability.

S — Structure

Entry: Confirm baseline knowledge of variables, ownership, references, and mutability.

Loop: Mentor seeds, probes, constrains, and verifies reasoning as the learner writes, explains, and tests the function.

Exit: Learner demonstrates correct parameter passing behavior, explains effects on variables, and can generalize to new examples.

Exercise 2

Designing a PARTS Contract — Teaching a Visitor About Your City

Design a PARTS contract that teaches a visitor about your city. The goal is not just to describe your city, but to learn how to design learning—intentionally, reflectively, and with clear structure.

By the end of this task, you should be able to independently create a coherent PARTS contract and explain why each part is designed the way it is.

The End