Prompt architecture cheat sheet

The specification framework from Artificial Leverage by Matthew Jefferies

The core principle

Prompts are specifications, not requests. The quality of AI output is bounded by the specificity of the input. A vague specification produces a vague result. A precise specification produces a precise result.

The specification test

> *"If I gave this instruction to a competent contractor who knew nothing about my context, would they produce what I need?" If no, your prompt is underspecified.*

Five predictable gaps in underspecified prompts

Audience	Who is this for? A summary for your CEO and one for your team are fundamentally different documents.
Format	Prose, bullet points, table, email, or slide outline? Without this, the model chooses for you.
Constraints	How long? What to exclude? What level of detail? Without constraints, output is comprehensive but unfocused.
Perspective	Optimistic or cautious? Advocate or present options? Assume context or explain from scratch?
Quality criteria	What does "good" look like for this specific output? If you cannot articulate it, the model cannot produce it.

The system prompt hierarchy

1. **Platform system prompt:** Highest priority. Safety layer from the AI provider. Cannot be overridden.
2. **Custom instructions:** Your persistent configuration. Shapes defaults across all conversations in a project.
3. **Conversation context:** What you have said earlier. Lower priority than system-level instructions.
4. **Current prompt:** Highest attention for task execution, but system prompts win on behaviour constraints.

Elements of effective custom instructions

Role and context: Who you are, your domain, your institutional language

Communication preferences: Language variant, formatting rules, tone calibration

Output defaults: Default length, structure, and format preferences

Domain knowledge: Assumed familiarity, terminology, key frameworks

Quality standards: Directness, evidence requirements, hedging rules

Building reusable prompt architectures

Templates: Reusable specifications you adapt for specific tasks. If you do it more than 3 times, build a template.

Chains: Sequences of templates for multi-step workflows. Research to analysis to briefing. Each output feeds the next.

Libraries: Collections of tested, refined templates organised by task type. Build over time, refine based on results.

Diagnosing specification failures

When output disappoints, the cause is almost always a specification gap. Diagnose before rewriting.

Output ignores constraints	Move constraints to the top of your prompt. Position effects apply: the model attends most to the beginning and end.
Output is generic	You provided format but not perspective or quality criteria. Add what makes this output different from a generic version.
Model hedges too much	Override explicitly: "State your recommendation directly. Do not present on-the-other-hand alternatives."
Format drifts in long output	Break long documents into section-by-section prompts rather than requesting everything in one pass.
Output contradicts system prompt	Check the hierarchy. Restart conversation and restate key constraints if context has introduced conflicts.