

BEYOND THE CLEVER PROMPT

Your Prompts Are Requirements Documents. Start Treating Them Like It.

The rush to put language models into production is quietly relearning every lesson requirements engineering already paid for.

Giovanni Leonardi

June 2025



Your Prompts Are Requirements Documents. Start Treating Them Like It.

Transformation · No. 349
June 2025

"A prompt is a requirements document that executes the instant you write it, against a system that will never tell you which requirement you forgot."

The Demo That Worked

Every team has now had this moment. A language model is wired into something that matters — a triage step, a drafting tool, a summary a customer will actually read — and in the demo it is flawless. The prompt has been polished over an afternoon; the outputs are crisp; the room is impressed. The feature ships. Three weeks later someone notices it is quietly doing the wrong thing on a small, stubborn fraction of cases, in a way nobody caught because nobody had written down what right was supposed to mean.

We are calling the polishing of that prompt "prompt engineering", and treating it as a new craft — a knack, a bag of tricks, something you pick up by feel. It is worth saying plainly what it actually is. A prompt that governs a production behaviour is a requirements specification. And we are writing these specifications with less discipline than we would have tolerated for a form field twenty years ago.

What A Prompt Actually Is

Strip away the novelty and look at what a working prompt contains. It states the intent of the task. It sets the scope — what is in, what is out. It lists constraints: tone, format, length, the things never to do. It implies acceptance criteria, because someone decided the demo outputs were good enough to ship. That is not a trick. That is the anatomy of a requirement, the same anatomy we have been teaching analysts for decades.

There is one difference, and it is the whole point. A traditional specification is read by a person who will notice when it is incomplete and come back with questions. A prompt is a requirements document that executes the instant you write it, against a system that will never tell you which requirement you forgot. The model does not raise a query against an ambiguous instruction; it resolves the ambiguity itself, confidently, with the most statistically plausible reading — which may not be yours. Underspecification, which used to produce a delay, now produces a silent, fluent, wrong answer.

The old failure was a specification nobody read until it was too late. The new one is a specification the machine reads perfectly and completes on your behalf – filling every gap you left with a guess you never saw it make.

The Old Mistakes, Wearing New Clothes

Anyone who has lived through a requirements-heavy programme will recognise the failure modes now reappearing under the banner of prompt engineering.

- Ambiguity. We are specifying behaviour in natural language, which is exactly as slippery as it always was. "Summarise concisely" means three things to three people, and a fourth to the model.
- Implicit assumptions. The most dangerous requirement is the one so obvious nobody states it. I have watched a support-triage prompt sail through two hundred hand-checked examples and then, in production, mislabel roughly one ticket in twenty-five – because the rule everyone in the room knew, that anything mentioning a regulator gets escalated, had never been written into the prompt. At four per cent, across tens of thousands of tickets a week, that is not a rounding error. It is a queue of real problems routed to the wrong place, and found late.
- No acceptance criteria. "It looked good in the demo" is not a test. A handful of happy-path examples is not coverage. We knew this about software; we are pretending not to know it about prompts.
- No version control. The prompt gets tweaked on a Friday to fix one complaint, and now nobody can say which wording is actually live, or what that change quietly broke elsewhere. Requirements taught us to manage change deliberately. Prompts are being edited in place, in production, by whoever has access.

None of these are novel problems. They are the founding lessons of requirements engineering, being relearned at speed because we gave the activity a new name and assumed the old lessons no longer applied.

"But Prompting Is A Conversation"

The strongest objection to all this deserves taking seriously, because it is half right. Prompting, the argument goes, is not specification at all – it is conversation. It is iterative, forgiving, cheap to change. The whole promise of working this way is that you escape heavy up-front requirements: you try something, see what comes back, and adjust. Insisting on rigorous specification misses what makes the tool powerful.

That is true – at the workbench. While a human is in the loop, reading every output and steering, iteration genuinely is the method, and imposing ceremony on it would be foolish. But the forgiveness is an illusion the moment the loop closes. When the model's output flows straight to a customer, or into a downstream system, or triggers the next step of an autonomous sequence, there is no longer anyone reading each result and catching the occasional wrong one. The prompt is now load-bearing. "Cheap to change" is

not the same as "safe to change" — and the gap between those two is exactly the distance a production system has to cross.

The One Move Worth Making

The correction is not heavier process. It is borrowing the small, cheap disciplines that requirements engineering already worked out, and applying them only where a prompt has become load-bearing.

1. Write the acceptance criteria before polishing the prompt — a set of real cases, the awkward ones included, each with the behaviour you expect. If you cannot say what right looks like, you are not ready to ship.
2. Name the implicit assumptions out loud. The regulator rule. The tone you would never use. The thing everyone knows. If it matters and it is not in the prompt, it is not a requirement — it is a hope.
3. Version the prompt and review changes like the specification changes they are. Not bureaucracy; just knowing what is live and what moved.

Done lightly, this costs an afternoon and spares you the quiet three-weeks-later discovery that erodes trust in the whole endeavour.

The uncomfortable truth for anyone selling prompt engineering as a shiny new discipline is that the people who will be quietly excellent at it are the ones who learned, often painfully, to write a good requirement. They already know that the gap between what you asked for and what you meant is where systems fail. The tool is new. The failure is not.