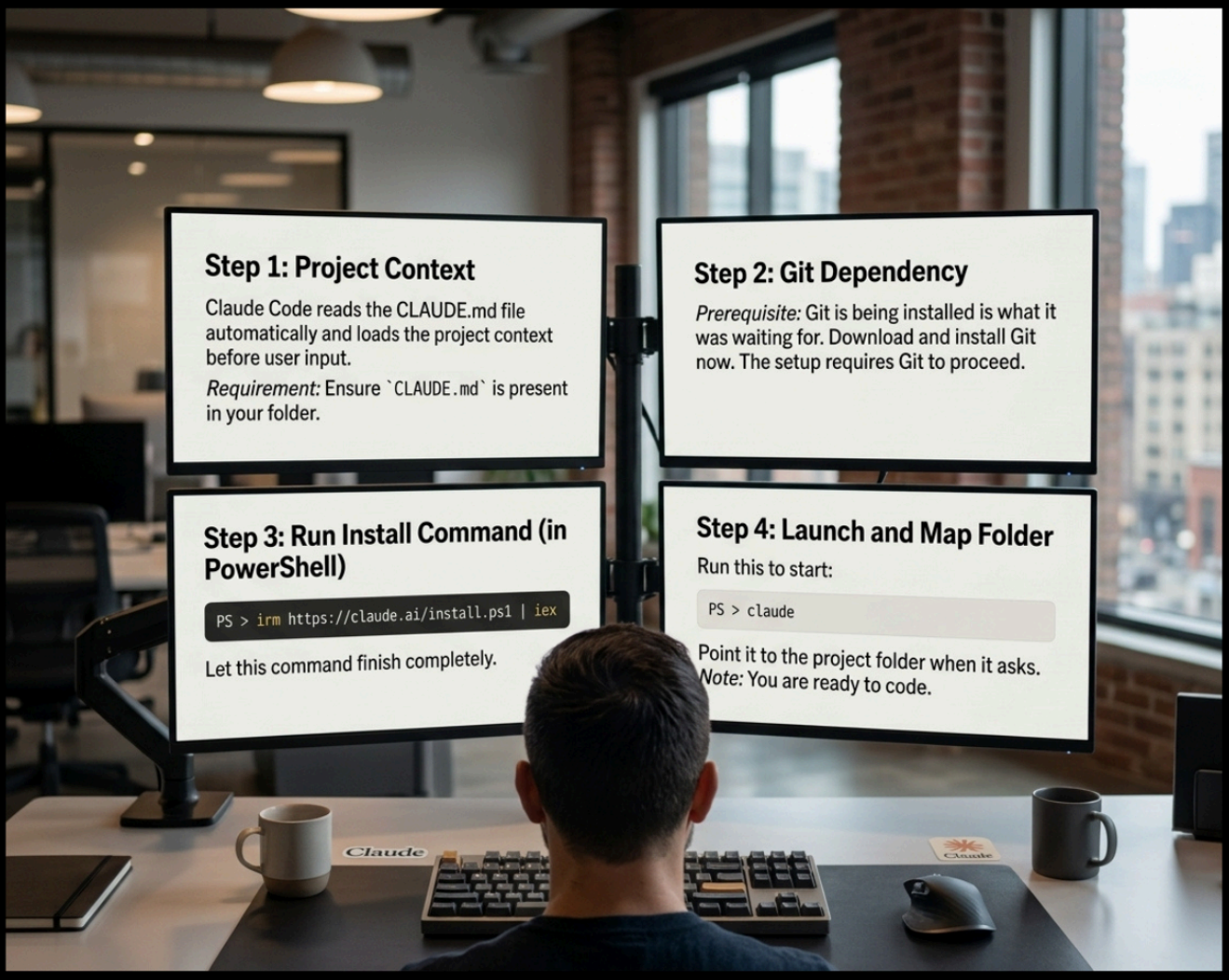
THE AI SPECTATOR

YOUR LENS INTO THE WORLD OF AI

THE INSTRUCTION GAP

Following Sequential AI Output as a Measurable Human Skill



THE BOTTLENECK HAS SHIFTED FROM PROMPTING TO EXECUTION

WORKING MEMORY CAPACITY DETERMINES WHO COMPLETES AN AI BUILD

FEWER THAN 20% OF ENTERPRISE AI PROJECTS ACHIEVE EXPECTED ROI

WHO GETS LEFT BEHIND WON'T BE WHO CAN'T PROMPT , IT'S WHO CAN'T FOLLOW THROUGH

By: David Borish, AI Strategist | The AI Spectator | June, 2026

Abstract

As agentic AI systems become capable of generating complete software builds, integration sequences, and multi-step deployment instructions from natural language input, the human bottleneck has shifted. This paper argues that the ability to follow sequential instructions returned by AI coding agents is a distinct, measurable cognitive skill that is currently underdeveloped and underrecognized across both individual users and enterprise AI programs. Drawing on psychological research into working memory, instruction-following behavior, and social compliance, it frames instruction-following capacity as a competitive differentiator in AI-assisted development and as a primary failure mode in enterprise AI rollouts. It proposes that organizations and individuals who treat this skill as trainable and measurable will outperform those who focus exclusively on prompting ability.

I. The Shift Nobody Named

When AI coding tools first gained serious traction, the dominant conversation was about prompting. Learn to write better prompts, the argument went, and you would get better output. That framing was reasonable in 2023. The models were capable but inconsistent, and the quality of a prompt had a direct and observable effect on the quality of what came back. Prompting skills were real, teachable, and consequential. Entire consulting practices, online courses, and job descriptions were built around the idea that prompt engineering was the frontier skill of the AI era.

By 2026, that framing is incomplete. Not wrong, but insufficient. The models have improved to the point where a reasonably clear natural language request produces usable output most of the time. Claude Code, GitHub Copilot, Cursor, and their peers can now generate multi-file applications, configure dependencies, and scaffold production-grade architectures from plain descriptions. Andrej Karpathy's concept of vibe coding, which he introduced in early 2025, captured something real: the primary role of the human in software development was shifting from writing code to directing an agent that writes code. The capability gap that made prompting so critical has narrowed substantially.

What has not narrowed is the execution gap on the human side. When an AI coding agent returns a sequence of steps, someone has to complete them. Create an account on this platform. Generate an API key with these specific permissions. Set this environment variable in this file. Run this command in the terminal in this directory. Paste this value here, not there. Confirm the connection before proceeding to the next step. Each instruction is discrete. Each is dependent on the previous one. And each requires a human to actually execute it, completely, before moving forward.

This is where builds break. Not because the AI produced bad code. Not because the prompt was poorly constructed. Because someone skipped step four, or half-completed step two, or misread which field the API key belonged in, or assumed a step was optional when it was not. The build fails. The developer spends hours debugging what turns out to be a configuration error introduced in the first fifteen minutes. The AI gets

blamed. The tool gets labeled as not ready for production. The actual cause, an incomplete instruction sequence executed by a human, goes unexamined.

The bottleneck has moved, and the field has not caught up to where it moved. This paper is an attempt to name the problem precisely enough to address it.

II. What the Research Tells Us

The psychology of instruction-following has been studied formally since the late 1970s, when researchers first observed that people consistently prefer to start using a new device rather than read the manual that came with it. That preference, it turned out, was not laziness or impatience. It was a function of how working memory operates under the load of sequential tasks, and it has been replicated across populations, settings, and instruction types for nearly five decades.

A 2020 commentary published in the American Journal of Pharmaceutical Education reviewed the accumulated literature and identified working memory capacity as the primary constraint on instruction-following performance. The argument is precise: holding a multi-step sequence in mind while simultaneously executing each individual step places measurable demands on working memory. When those demands exceed capacity, people start sequences and do not finish them. They execute step one and step two, lose the thread at step three, and move on under the impression that they are done. The failure is not motivational. It is cognitive, predictable, and patterned.

Working memory is not a fixed trait. It varies across individuals and it degrades under specific conditions: time pressure, environmental distraction, emotional stress, unfamiliarity with the domain, and cognitive load from parallel tasks. All of these conditions are present in the environments where people are most likely to be following AI-generated instruction sequences. Someone setting up their first API integration is under all five simultaneously. They are under mild time pressure because they want to see the build work. They are in an environment with notifications and interruptions. They are dealing with the stress of unfamiliar tooling. They are in a domain they do not know well. And they are holding a mental model of the larger project in parallel with the specific steps in front of them. The research predicts, reliably, that instruction sequences will be incompletely executed in these conditions. Practitioners know this from direct observation.

Susan Gathercole's research at Cambridge documented the breakdown pattern with particular clarity. Children with lower working memory scores consistently started instruction sequences but failed to complete them. The failure was not random. It was patterned: early steps got executed, later steps did not. The sequence fell apart at the back end, not the front. The same architecture applies to adults working through unfamiliar technical sequences. Starting is easy. Maintaining the sequence through to full completion is where working memory capacity becomes the constraint.

A 2016 study by Jaroslawska, Gathercole, Allen, and Holmes identified a practical intervention with a strong effect size. Physically performing each step at the moment of instruction, rather than reading or hearing the full sequence and then attempting to

execute it, significantly improved both retention and completion rates. The mechanism is enactment: doing the step encodes what comes next. When someone reads an eight-step API setup guide and then attempts to execute all eight steps from memory, they are placing the full sequence load on working memory at once. When they read step one and immediately do step one before reading step two, the load is distributed across execution rather than concentrated in memory. Completion rates improve substantially under the enactment model.

A 2015 study published in Scientific Reports from researchers at the Chinese Academy of Sciences and the University of Leeds examined instruction-following under varying input modalities and found that demonstrated instructions produced significantly better recall and completion than spoken or written ones. When each step is shown rather than described, the human encoding the instruction has a richer representation to draw on during execution. This finding explains a pattern that every developer who has worked with non-technical colleagues has observed: written setup documentation produces higher error rates than video walkthroughs of the same steps, even when the written documentation is more precise and complete. The modality matters, not just the content.

Stanley Milgram's obedience research adds a social dimension that is underappreciated in this context. Milgram demonstrated across nineteen variations of his core experiment that instruction-following is situational rather than dispositional. People are not generally compliant or generally non-compliant. They follow instructions at higher rates when an authority structure is present and enforcing completion. When that structure is removed, when the authority figure leaves the room or communicates by telephone rather than in person, compliance drops measurably and predictably. An AI coding agent returns a numbered list of steps and waits. There is no authority in the room. There is no checkpoint structure built into the process. Milgram's data predicts exactly what practitioners observe: a significant portion of people do not complete the sequence as specified.

III. How Prompting Captured the Conversation and What It Missed

The rise of prompt engineering as the defining skill of the AI era followed a logic that made sense at the time. In 2022 and 2023, the quality differential between a well-constructed prompt and a poorly constructed one was large and visible. A good prompt produced coherent, useful output. A bad prompt produced something that missed the point or required multiple rounds of correction. The connection between prompting skill and output quality was direct enough to observe without instrumentation. It felt like a skill because it produced measurable, visible results.

The industry responded accordingly. Prompt engineering courses multiplied across online learning platforms. Job postings appeared for prompt engineers with salaries that reflected genuine market demand. Publications ran guides on how to structure prompts for different use cases. OpenAI, Anthropic, and other labs published prompt engineering documentation. The skill became legible, teachable, and credentialed in the informal way that new technical skills become credentialed before formal certification exists.

What this framing embedded was an assumption about where value was being created in the human-AI exchange. Prompting is an input-side skill. The human does something, and the quality of what they do determines the quality of what comes back. That is a satisfying model because it places human agency at the creative moment. The human is not passive. The human is making decisions that shape the output. The skill is upstream, visible, and feels like authorship.

Instruction-following is an output-side skill. The agent does something and hands it back. The human's job at that point is execution, not creation. There is no authorship involved. There is no creative decision. There is a list, and the list either gets completed or it does not. This does not feel like a high-status skill. It feels like administration. The implicit assumption in how the field has talked about AI development is that anyone can follow a list. That assumption is contradicted directly by the research, but it is persistent because it is intuitive.

The confidence problem compounds this. When someone fails to follow an instruction sequence correctly, they almost never attribute the failure to their own instruction-following behavior. They attribute it to the instructions. The tool was confusing. The documentation was unclear. The AI made an error. The setup process was poorly designed. All of these attributions may be partially true. But the research on working memory and multi-step task completion shows that a significant portion of sequence failures are caused by incomplete human execution of correctly specified steps, not by errors in the steps themselves.

The prompting-as-primary-skill narrative also developed during a period when AI tools required more prompting sophistication to produce useful output. As models have improved, that requirement has decreased. The same output that required careful prompt construction in 2023 can now be produced with a much less precise request. The skill ceiling for prompting has risen, but the floor has also risen, meaning that adequate prompting requires less investment than it once did. Instruction-following capacity has not changed in either direction because it is constrained by human cognitive architecture, not by model quality. The relative importance of the two skills has shifted, and the field has not updated its framing to reflect that shift.

IV. The Enterprise Failure Pattern

Enterprise AI programs have invested heavily in the input side of the human-AI exchange. Model selection processes are rigorous. Vendor evaluations are thorough. Prompt engineering training has become a standard component of AI adoption programs at large organizations. Security and compliance reviews are extensive. The organizational infrastructure built around getting AI tools selected, approved, and deployed is substantial.

Almost none of this investment addresses what happens after deployment, at the moment when a human being has to follow the instruction sequence that connects an AI tool to the systems it needs to work with. That moment is where most enterprise AI implementations break down, and the breakdown is consistently misdiagnosed.

The misdiagnosis follows a predictable pattern. An AI tool is selected and approved. An implementation team is assembled, typically comprising a mix of technical and non-technical staff. The AI tool generates an integration sequence: configure these permissions, set up these API connections, establish these data pipelines, confirm these environment variables. The team works through the sequence. Some steps get completed correctly. Some get partially completed. Some get skipped because they appeared optional or were not understood. The integration fails. The failure gets diagnosed as a technical problem, an incompatibility between systems, a gap in the tool's capabilities, or a flaw in the implementation plan. The actual cause, incomplete execution of a correctly specified instruction sequence, is not examined because nobody is looking for it.

Boston Consulting Group research has estimated that fewer than 20 percent of enterprise AI projects achieve their expected return on investment. This statistic is cited frequently in discussions about AI adoption, and the explanations offered typically focus on model limitations, data quality issues, change management failures, and strategic misalignment. Instruction-following capacity as a failure mode does not appear in these analyses, not because it is not present, but because it has not been named or measured.

The specific mechanics of enterprise instruction-following failures differ from individual failures in important ways. In an individual context, one person is executing a sequence and the failure is localized. In an enterprise context, instruction sequences are often distributed across team members, creating coordination failures in addition to individual execution failures. Person A completes steps one through four and hands off to Person B, who picks up at step five without knowing that step three was incompletely executed. The build fails at a point that is six steps removed from the actual error. The trace is difficult and time-consuming.

Enterprise environments also introduce the authority and proximity effects that Milgram's research identified. In a team implementation, nobody has clear authority over instruction-sequence completion. The project manager is tracking milestones, not step-level execution. The technical lead is focused on architecture, not configuration. The individual contributors are executing steps without a checkpoint structure that confirms completion before advancement. The social conditions that Milgram showed to be necessary for reliable instruction-following are absent from most enterprise AI implementation processes.

The organizations that have succeeded in AI implementation share a common characteristic: they treat it as a procedural discipline, not a technical one. They build verification checkpoints into integration sequences. They require confirmed completion of each step before the next step is presented. They assign clear authority over sequence execution to a single person rather than distributing it across a team. They debrief failed implementations by examining step-level execution rather than diagnosing tool failures. These are not sophisticated interventions. They are applications of what the instruction-following research has shown for decades.

V. Who Gets Left Behind

The workforce displacement argument about AI has focused primarily on which jobs will be automated and on what timeline. The Exponential Replacement Curve framework I published in May 2025 mapped that displacement across three waves based on task susceptibility to AI completion. The instruction-following dimension adds a layer to that analysis that the automation timeline alone does not capture.

As AI coding agents become capable enough to build production software from natural language, the technical skills that once separated developers from non-developers are compressing. The floor is rising. Someone with no programming background can now produce a working application if they can describe what they want with sufficient clarity and follow the setup sequence the agent returns. That is a genuine shift in who can participate in software creation, and it is happening faster than most workforce planning has accounted for.

But the rising floor does not eliminate the requirement for sequential execution. It shifts what the execution involves. The work is no longer writing code. The work is completing the non-code steps that connect working code to a working system. API authentication, environment configuration, account provisioning, permission management, service integration. These steps are not technically demanding in the way that writing a sorting algorithm is technically demanding. But they are sequentially demanding. They require the same multi-step working memory load that the research has shown to be a genuine differentiating constraint.

The people who will be left behind as AI coding tools mature are not necessarily the ones who cannot write good prompts. They are the ones who cannot complete what the agent hands back. The correlation between technical background and instruction-following capacity is weaker than it might appear. Experienced developers have better domain knowledge, which reduces the cognitive load of unfamiliar steps. But domain knowledge is not the same as instruction-following discipline. Many experienced developers have developed habits of improvisation that produce exactly the kind of partial execution that breaks AI-assisted builds.

Non-technical users, by contrast, often follow instructions more completely because they have no basis for deciding which steps to skip. They do not know enough to improvise, so they execute the sequence as specified. This makes them, in some respects, better suited to AI-assisted development workflows than developers who have spent years building shortcuts into their practice.

The workforce implication is that instruction-following capacity is not distributed along the lines that hiring and advancement decisions currently assume. Organizations that assess and develop this skill explicitly will access talent pools that traditional technical hiring filters out. Individuals who recognize this shift and deliberately develop their instruction-following discipline will build things that their technically more sophisticated peers cannot, because those peers will keep breaking builds by skipping steps they think are unnecessary.

VI. What Good Instruction-Following Practice Looks Like

The research provides a clear enough foundation to specify what good instruction-following practice looks like in an AI-assisted development context. These are not speculative recommendations. They are applications of findings that have been replicated across populations and settings.

The first principle is enactment before advancement. Do not read an instruction sequence in full before beginning execution. Read step one, execute step one completely, confirm completion, then read step two. This is the enactment model that Jaroslawska and colleagues demonstrated produces significantly higher completion rates than batch reading followed by batch execution. It feels slower. It is not slower when measured against the time cost of debugging a failed build caused by an incompletely executed step.

The second principle is explicit verification. Completion of a step is not the same as the belief that a step was completed. Many instruction-following failures occur because someone executed a step incorrectly and did not verify the result before advancing. Each step in an AI-generated sequence should have an explicit verification action: confirm that the API key was accepted, confirm that the environment variable is present in the correct file, confirm that the service is returning the expected response. When the AI coding agent does not specify verification steps, add them.

The third principle is sequence integrity. Do not deviate from the specified sequence because a step looks familiar or optional. The working memory research shows that the steps most likely to be skipped are the ones in the middle of a sequence, after the initial steps that feel essential and before the final steps that feel conclusive. Middle steps are where the cognitive load is highest and where the deviation temptation is strongest. They are also where the most consequential errors occur, because later steps are built on their completion.

The fourth principle is single-thread execution. Do not work through an instruction sequence while also monitoring other tasks, responding to messages, or holding parallel processes in working memory. The research on working memory degradation under cognitive load is unambiguous: additional demands on working memory reduce the resources available for sequence retention and execution. If the instruction sequence matters, it deserves undivided attention until it is complete.

The fifth principle is debrief on failure. When a build fails, examine the instruction sequence execution before diagnosing tool failure. Walk back through each step and confirm that it was completed as specified, not as interpreted. This is cognitively uncomfortable because it requires acknowledging that the failure may have been caused by incomplete human execution rather than a tool error. Organizations that institutionalize this debrief will systematically improve their instruction-following capacity. Organizations that default to tool failure diagnoses will repeat the same failures indefinitely.

For organizations, the structural implication is checkpoint architecture. Instruction sequences above a certain complexity threshold should not be executed without checkpoints that require confirmed step completion before advancement. This is standard practice in aviation, medicine, and nuclear operations. The checklist research that Atul Gawande documented in surgery and intensive care units is directly applicable here. The cognitive mechanisms are the same. The intervention that works is the same.

VII. What Comes Next

This paper is an initial framing, not a completed study. The empirical foundation it draws on comes from research conducted in contexts, classrooms, clinical settings, laboratory experiments, that are adjacent to but not identical to AI-assisted software development. What the field needs is direct measurement in the actual context.

Specifically: task completion rates across user populations working with AI coding agents, measured at the step level rather than the outcome level, broken down by instruction sequence complexity, user background, environmental conditions, and instruction modality. We need to know whether instruction-following capacity predicts successful AI-assisted builds more strongly than prompting quality does. The working memory literature suggests it should. Practitioners in this space will recognize the pattern from direct experience. But a rigorous measurement study would move this from informed observation to quantified evidence, and quantified evidence is what changes organizational practice.

Organizations investing in AI capability development should add instruction-following assessment to their evaluation frameworks alongside prompting evaluation. Assess how people perform on multi-step technical sequences under realistic conditions. Identify where in those sequences completion breaks down. Design training that applies the enactment and checkpoint findings from the working memory research. Measure improvement over time.

The AI field has spent two years optimizing the input side of the human-AI exchange. That work was not wasted. Prompting matters, and it will continue to matter as the complexity of what we ask AI systems to do increases. But the output side, the human execution of what AI systems return, is where the work is now. The models are ready. The question is whether the humans who work with them are equipped to complete what they are handed. The instruction gap is real, it is measurable, and it is addressable. Naming it correctly is the first step.

References

Dunham, S. et al. "The Psychology of Following Instructions and Its Implications." *American Journal of Pharmaceutical Education*, 2020. https://pmc.ncbi.nlm.nih.gov/articles/PMC7473227/

Gathercole, S.E. and Alloway, T.P. *Working Memory and Learning: A Practical Guide for Teachers.* Sage Publishing, 2008.

Jaroslawska, A., Gathercole, S.E., Allen, R., Holmes, J. "Following instructions from working memory: Why does action at encoding and recall help?" *Memory and Cognition*, 2016. https://pmc.ncbi.nlm.nih.gov/articles/PMC5085979/

Milgram, S. *Obedience to Authority: An Experimental View.* Harper and Row, 1974.

Yang, T. et al. "The influence of input and output modality on following instructions in working memory." *Scientific Reports*, 2015. https://pmc.ncbi.nlm.nih.gov/articles/PMC4669483/

Waterman, A.H. et al. "Do actions speak louder than words? Examining children's ability to follow instructions." *Memory and Cognition*, 2017. https://pmc.ncbi.nlm.nih.gov/articles/PMC5529483/

Gawande, A. *The Checklist Manifesto: How to Get Things Right.* Metropolitan Books, 2009.

Borish, D. "The Open-Prem Inflection Point V3." The AI Spectator, April 2026. https://www.davidborish.com/open-prem-inflection-point-v3

Borish, D. "The Exponential Replacement Curve." The AI Spectator, May 2025. https://www.davidborish.com/the-exponential-replacement-curve

---

David Borish is an Enterprise AI Strategist, NYU Guest Lecturer on AI, and author of the Open-Prem Inflection Point and Exponential Replacement Curve frameworks. He writes at The AI Spectator.