The Five Cs

Adam Daw

2026-04-25

Contents

The Five Cs

As those of you who have known or been following me for a while are aware, I stepped away from full-time development for a while to focus on my mental health and spent more time teaching and doing research, as well as exploring prose writing. This changed last year, and I've been working on several projects over the more recent period that have given me a lot of insight into how the machine learning and "Agentic AI" space has changed over the course of several years. It's been a shock to see how rapidly things are ramping up, and there's an energy in the software development space that I haven't seen since the early days of Web 2.0, or before it around the turn of the millennia. Recently it seems that two camps have swiftly formed — most programmers I talk with either have a story, often several, about something they built or shipped recently because an agentic coding tool helped them do it, or a growing fear that their opportunities are going to evaporate as new tools render their role defunct.

This sentiment isn't new, and like the energy that seems to resurge every decade or so, is often exacerbated by stories and media segments that benefit more from engagement than education. On the other hand, new tools are present, and are, as I see it, genuinely useful. I think they will keep getting better. In the past 18 months I have shipped work I would not have shipped without one of them on hand. When that has happened, it has nearly always been because I attended carefully to a few specific things, and got out of the model's way on the rest.

The hardest part of this, though, is identifying *which* things, exactly, are worth that careful attention — and that campaign of conversation and discovery has been less precise than the technology itself has been productive. The reflexive answer, when it comes, is creative work, taste, judgement, "the human element." Those answers are not wrong. They are also not specific enough to be useful, and the moment a colleague asks how to *practise* them, the conversation thins.

What I would like to offer is a working map of where, in my experience, that careful attention pays

for itself most clearly. The map has five places on it — five categories of effort where the human surpasses the tool, and where, as a consequence, the tool itself becomes a real accelerant of the work rather than an adjunct to it. I call them **the five Cs: Constraints, Context, Curation, Conceptualization, and Creativity.** Each is a discipline in its own right, and each is also, I believe, worth examining on its own terms.

The argument that links the five — to anticipate it briefly here so it sits and percolates while the body of the piece works — is one of structural asymmetry. Like all its predecessors, really, a modern model's reach is bounded by its training distribution: extraordinary in volume, mapped, predictably densest at the centre and thinning at the edges. The human's reach is bounded by lived experience: narrower in volume, but uncatalogued, historically situated, and through the inevitable associations that come from the jumble of neurons and glial cells we call a brain, capable of pulling in the parallel from another domain at the moment the model would have produced its safe, median guess. Almost everything that follows is some specific way of saying that the categories where the human surpasses the tool are the categories the training distribution cannot supply.

The piece takes the five Cs in turn. We begin with **Constraints**, because it is the most concrete, and because every other C rests, to one degree or another, on an understanding of what constraints are and how they narrow a space of possible outputs. From there we turn to **Context**, which narrows the space of inputs a tool is working from. **Curation** follows, as the discipline that decides *which* context is worth paying for — in the acute moment of a single session, and durably across the months and years a team keeps building. **Conceptualization** then takes up the human work of framing a problem well enough that the agent can act on it usefully. We close on **Creativity**, which is the place the other four feed into, and the part of the collaboration most often reflexively reserved for humans without close examination.

None of this is to say that AI tools are limited, nor that our current capabilities are the end of the story. I believe these tools will continue to improve, and I am genuinely excited to see where they go. What I would offer instead is that the most effective use of these tools today — the use that turns them into a real accelerant of the work, and not merely a clever adjunct to it — comes from attending carefully to the categories of effort where the human surpasses the tool. The five Cs are my working map of those categories.

Constraints

Constraints are the most concrete of the five Cs, and the easiest to demonstrate. The accurate answer to the question *what is a generative model doing* — that it navigates a high-dimensional space of learned relationships between concepts, compressed into billions of parameters — is true but not useful for understanding how a person applies or interacts with one. A working picture is what lets you start making good decisions about how to use these tools, and the somewhat reductive stand-in I have found works best for Constraints starts with a multiplication table.

Imagine a multiplication table that runs on forever. Any number of factors, any product, billions of examples stretching off in every direction. That vast unrestricted table is our picture of the space of things a generative model could produce.

Now let's say you're looking for a specific calculation. You don't know exactly what it looks like, but you know that your product has to be 8. That alone doesn't narrow things much — there are endless paths to 8. You know roughly what it should look like: $x \times y \times z = 8$. This is where constraints come in.

Say you filter the table down to integer factorisations using three numbers. The field is already much smaller, but you still have options: $1 \times 1 \times 8$, $1 \times 2 \times 4$, $2 \times 2 \times 2$, and their permutations. All of them arrive at 8. Now add one more constraint — each number can only be used once. That rule has nothing to do with the product; $2 \times 2 \times 2$ gets you to 8 just as faithfully as $1 \times 2 \times 4$ does. What the rule does is eliminate answers we couldn't articulate a reason to reject. You land on $1 \times 2 \times 4$.

That, I would propose, is what constraints really do when we're directing an agent. They narrow the neighbourhood of outputs we'll accept.

And none of this works without the product. "Must equal 8" is the most important piece of the example, because it's the thing that tells you you've arrived. When we're writing code with an agent, the product is the test, the spec, the acceptance criterion — the defined endpoint. Constraints narrow the search; the endpoint tells you when to stop.

Clear requests with defined endpoints, and the right constraints around them, are how we steer an agent toward the code we actually want. A single correct answer is a fool's errand in a space this big. What we can do instead is draw a small enough neighbourhood of right answers that whichever one the agent produces is one we can use.

It's an old idea dressed in new clothes. In the 1960s the Oulipo writers — Raymond Queneau, Georges Perec, and company — built a whole literary movement on the premise that constraints are a creative force. Despite their origins as a subcommittee of the *Collège de 'Pataphysique* and its absurdist, parodic nature, there is real truth to the ability of constraints to generate new ideas. Perec famously wrote a novel without ever using the letter *e*. The rule narrowed the neighbourhood of novels he could write, and a remarkable book came out the other side. There is some irony in discussing Oulipo when exploring what is reductively a "next word generator," but I'm not sure that's a bad thing.

Here's another example, still a step removed from code but easier to picture. Suppose I wanted a transformer to produce the sentence "The quick brown fox jumped over the lazy dog." I might tell it that each letter of the alphabet must be used at least once, that the sentence should include two animals (one in the subject, one in the predicate), and that both animals should be briefly described with at least one adjective.

Those constraints could produce "The quick brown fox jumped over the lazy dog." They could just as reasonably produce "My big fluffy cat quickly pawed an anxious zebra jogging over the weed patch," or "The vexed quokka furtively clawed a jumpy wombat snoozing near the glazed pier." All three satisfy the rules. All three live in the neighbourhood I described.

If I add one more constraint — "the sentence should have fewer than forty characters, excluding whitespace" — the neighbourhood collapses dramatically. Whatever comes back will be close to what I had in mind.

The endpoint tells the agent when it has arrived; without it, no amount of constraint will do the work you are hoping it will. The constraints themselves shape the neighbourhood of answers you will accept, so that any answer from inside that neighbourhood is a usable result. With both in place, steering an agent feels less like a contest and more like a conversation.

Context

Constraints, as I have just described them, narrow what comes out of an agent. They say nothing about what goes in — and what goes in is *context*. The meaning of that term has drifted far enough across the last few years that I'm not sure we're always talking about the same phenomenon when we use it. So, in the vein of the late and great Mortimer J. Adler, let's come to terms. I'd like to offer three meanings I see in circulation, lay each of them out briefly, and then argue that the third — the one easiest to skim past — is where most of the challenge in using AI well actually lives.

The first and most technical meaning is the *context window*: the literal set of tokens a model can hold in view at any given point in time. When an AI User says that the context is blowing up, they usually mean this one. It's the nuts and bolts definition: crisp, bounded, measurable, and almost always the first thing a new user of these tools learns about.

The second is what I'd refer to as the *operating context*: the configuration the model finds itself in. System prompts, persona framing, available tools, temperature, the rolling conversational history that makes turn two different from turn one. This is the fuzziest of the three, and I'm still sharpening how I think about it, but I find the distinction worth keeping. "What tokens are in view" and "what the model has been told about its job" are not the same thing, even when they share a lot of commonalities.

The third is the meaning I'm most interested in here: the broad sense in which context is the body of institutional and subject-matter knowledge the human operator brings to the problem but that the AI agent may or may not be aware of. This is context in the sense of what a new hire often struggles with — the *why* of a system being the way it is, *who* the stakeholders actually are versus who they should be, the *what* of every solution that has already been tried, and *which* decisions are off-limits for reasons you won't find in the code. A model can fake subject-matter expertise in domains that are well-covered in its training: the rules of chess, the grammar of French, the syntax of Kubernetes manifests. It will struggle on institutional context almost by definition, because institutional context is not in the training set; it's in the meeting last Thursday or stuck in a sticky note on a whiteboard from six months ago.

That distinction is worth explicitly outlining. Subject-matter context and institutional context are both better provided by the human, but they have different half-lives and different failure modes. A model can plausibly approximate subject-matter knowledge on a well-trodden domain; it cannot approximate why your team decided not to adopt Redis in 2023.

Here is the part of the picture I most want to highlight. Even setting aside the human-only third meaning, context is expensive. The per-token cost of inference has fallen a long way and will keep falling, but that alone does not make context cheap. Long contexts cost time — first-token latency climbs, and content generation slows. They cost attention, too; a well-known 2023 paper, "Lost in the Middle," showed that model reasoning degrades when relevant facts are buried in a long prompt, even when those facts are demonstrably present. Free context isn't free. The model pays in diluted attention whether or not anyone is billing for it.

And the harder problem is not the cost, but rather the selection. Identifying *which* context is worth paying for — which instructions, which examples, which facts, which slices of the last three months of incident history — is a judgement call that depends on the task, the domain, the moment, and the stakeholders. Retrieval systems approximate this with similarity scores, which are useful, and might often be sufficient for generic work, but are brittle at the points that matter most: that territory of expertise the third meaning of context most often occupies.

What does a human do efficiently that a retrieval pipeline does clumsily? In this context, I would say that they better decide what is earning its keep. They might know that one legacy lookup table still matters and that the migration doc from the spring has already gone stale, but the person who was meant to upload the new one went on leave and no one has had a chance to update it. They know that the one-page project brief the client actually gave sign-off on over lunch last week is worth more than the forty-page architecture document that sat in their inbox. A working engineer, given five minutes, can often compress into a paragraph the context a vector store chasing the same problem would approximate across tens of thousands of tokens — and still miss the load-bearing nuance.

This connects directly to the previous section. Constraints narrow the neighbourhood of acceptable outputs; Context narrows the neighbourhood of acceptable inputs. Both are crucial at different points in the agent's operation. A good prompt is a compact, well-chosen slice of context. A bad prompt leaves both neighbourhoods — the input and the output — far too wide.

There is one more thread worth pulling, briefly. A 2022 paper from Berkeley by Snell, Klein, and Zhong ("Learning by Distilling Context") showed that you can fine-tune a model to behave *as if* it were still seeing a prompt, without the prompt being present at inference. Context, in that construction, becomes weight (in the AI sense of the term, not a metaphysical one). That complicates the neat picture of "context is what you pass in" — the boundary between passing and baking in is permeable. However, I would argue that it doesn't undo the argument: if context can be distilled into the model, the scarce resource simply shifts one layer up — the judgement of which context is worth making permanent. You still have to choose what to teach, as well as someone doing the choosing and teaching.

Two things hold across all three meanings. Context is expensive — in dollars, in latency, and in the model's attention — and that expense is not disappearing just because per-token prices keep falling. And identifying *which* context is worth paying for is a judgement call that does not cleanly reduce to similarity search; the person who can make that call is adding real value, and in many settings is doing it more efficiently than anything else in the loop.

The prompt, the spec, the short paragraph written at the top of the file — these are the places that work shows up. Get those right and the agent spends its tokens on the right problem. Get them wrong and it spends them beautifully on the wrong one.

Curation

A quiet ritual precedes every real session I have with an agentic coding tool. I gather files, trim what I don't need, and probably surface a note I made sometime the week prior because I think the agent is going to need it. I remind myself which design I'd sketched out was superseded three weeks ago and which still has relevance. The first time I paid attention to what I was actually doing, it felt administrative, like planning for a trip the week before, but further reflection showed otherwise. This preliminary work is where a significant portion of the real decisions about the task are getting made.

Selection, not cost, was the harder of the two problems I named in the last section. The material cost of context falls a little further with every inference release; deciding *which* slice of institutional and subject-matter knowledge earns its place in the prompt this time does not. Identifying that slice is a judgement call. The variables are familiar — task, domain, moment, and the people involved — but the call is harder than the variables suggest, because they all move at once. It is a discipline,

and I believe that it has a name, though in AI discussions I've rarely seen it referred to specifically in this manner. The term I'd like to use is *curation*.

Curation is *at least* two different things, though both meanings share a verb and I think both are worth keeping. There is the **acute** curation of a single session's working set: which files are open, which notes are surfaced, which past conversation is carried forward, which piece of ambient context is worth passing into the prompt right now. And there is the **durable** curation of a long-running knowledge base: the body of docs, specs, rev histories, architectural decision records, CLAUDE.md files, institutional handbooks, and everything else an agent (or a human colleague) could reach into over the months and years a team keeps working together. One discipline operates on the time horizon of a single afternoon. The other operates on the time horizon of a career, or in the case of some long-running organisations, several careers. I would propose both are equally important, but they are definitely not the same activity, nor really the same skill.

The acute variety is the one most practitioners already do intuitively, even if they haven't given it a name. It's the ritual I described at the top. A working programmer preparing to hand a task to an agent does a small amount of deciding-before-prompting that has surprising consequences: what is in the context window will be used, what is not will be ignored, and what is in there but irrelevant will dilute attention and quietly confuse the model. Acute curation is the practice of being deliberate about the working set. Paying attention to what goes in, and maybe more importantly, what comes out.

In practice, the acute curator spends more time removing than adding. The most frequent action is taking a thing out, or declining to let it in to begin with. If I, returning to an example from the previous section, have forty pages of architecture doc and one one-page brief the client actually signed off on over lunch, the brief usually wins and the architecture quietly exits. If my vault is full of notes I captured three years ago when I was solving a different problem, those notes will gently pull the agent in that older direction unless I specifically narrow the field. Restraint, not abundance, is the working posture. Once again, we return to constraints as a guiding principle.

The durable variety is the one that most interests me, because it is where the long-tail human advantage concentrates. A well-tended knowledge base is a compounding asset — every interaction an agent has with it improves slightly because the inputs have been maintained. A neglected one is an attic: full of things, not quite useful, not quite retrievable, and psychologically oppressive in a way you can't put your finger on until you try to work inside it. As anyone who has dealt with clutter can attest to (or maybe not, and this is just my personal neuroses), extraneous content carries metaphysical weight by its mere presence. We can try to ignore it, actively and passively, but no success is complete.

The maintenance is labour, and I think it's worth saying so plainly. New information arrives daily, and most of it is not worth keeping. Old information ages, sometimes gracefully, sometimes into actively misleading residue. Decisions that looked sound six months ago turn out to have been the right call for the wrong reason, or the wrong call for the right one, and which is which needs to be recorded and the documentation properly categorised. The person doing durable curation is deciding what to add, what to revise, what to deprecate, and what to delete outright — and doing so against a fundamental uncertainty about which of those choices will matter later.

Chesterton's fence applies, of course. Removing something you don't understand is usually worse than leaving it in place; the fence is there for a reason that is probably not obvious to you, and pulling it down is how you discover the reason the hard way. On the other hand, a knowledge base that never cuts is potentially worse. The discipline is the ongoing work of deciding the difference —

a forward-looking judgement that is always partially wrong, made against incomplete evidence, with consequences that are difficult to measure except in retrospect. These are the types of decisions that are almost impossible to reduce to algorithm or decision tree, and are unlikely to lose their importance over time.

It's worth being clear about why this is skilled labour rather than simply tidying, and after significant thought, three reasons come to mind. First, the judgement is predictive: curation asks "will this matter?" rather than the significantly easier "did this matter?", and the asking must happen before the evidence has arrived. Second, the signal-to-noise ratio on fresh input is terrible, and deciding quickly under the pressure of that ratio is a wholly distinct skill from that of deciding slowly under ideal conditions. Third, editorial judgement is domain-specific. A librarian for a legal practice is not a librarian for a startup codebase; the things worth keeping in one are not the things worth keeping in the other. Curation is contextual — and we have just come from a section on context, so that shouldn't be a surprise.

A concrete example from my own practice. "claude-mem", a tool I've spent time using after recommendation from a respected peer, curates session memory for agentic coding tools. It does both disciplines at once, though I don't think I quite appreciated that until I began writing about it here. On the acute side, at the end of a session it decides which pieces of the conversation were meaningful enough to carry forward, and discards the rest. On the durable side, it maintains a cross-session record of what has mattered over weeks and months of interaction, allowing a future session to pick up on a level of ambient familiarity that would otherwise require re-explanation each time. What it decides to keep, emphasise and store isn't always what I agree with, and I end up keeping my own notes primarily due to the example it provides and the differences in our decision-making.

What makes the tool interesting to me, and what suggests the larger argument I've been making, is that the load-bearing work in both cases is the *selection*, not the storage. Storage is easy and cheap. As we've discussed, deciding what to store, what to drop, and what to surface later is where the discipline lives, and so far, the human curator's instincts tend to outperform any automated heuristic I've come across, though "claude-mem" is closer than most, I'll admit. Tools can assist — tagging, similarity-based surfacing, retention policies — and they already do. The *what is worth keeping* question, however, has not yielded to automation yet, and I'm not sure it ever fully will.

I've been describing what I referred to earlier in the narrow sense of a knowledge base: the structured store — docs, repos, vaults, memory systems, the artefacts you can list. There is a broader sense I want to acknowledge before closing, even if it is not the main subject of this section. Curation as a discipline extends past the structured store into organisational memory, unwritten conventions, the reasons certain decisions were made the way they were, the incident you only learned about over drinks with a departing colleague. The agent cannot reach that layer. The human can. If this article has a through-line, I'd argue it runs through exactly that fact: the disciplines where the human's edge over the tool is sharpest are the ones whose raw material the tool's training never gathered, and curation — especially durable curation — is where that edge most reliably shows up.

Two horizons, then. The acute and the durable. Both pose the same question — *what is worth keeping?* — and both have so far been answered better by a person who knows the work than by anything automated. Tools can assist, and they already do; the *what is worth keeping* question itself has not yet yielded.

Ultimately, I would picture curation as two different types of pruning — either of a bonsai, or of an orchard.

Conceptualization

There is a question that quietly precedes the others in this work applied to each project: namely, *what problem are we solving?* For the human in the loop working with an AI agent, deciding it is at least as much of the work as anything that follows.

Across constraints, context, and curation, I have been treating the problem itself as something the human had already settled before turning to the agent. We narrow the neighbourhood of acceptable outputs; we narrow the inputs the model has in view; we curate which slice of institutional knowledge rests as the foundation of the project. All of this presumes a problem statement to point those processes at. The discipline of producing that statement — of choosing the level of abstraction the conversation will operate at, of deciding whether the apparent problem is in fact the real one — is what I would like to refer to as *conceptualization*.

The move is not a new one. George Polya, in *How to Solve It* (1945), opened with the questions that come before solving: *what is the unknown? what are the data? what is the condition?* Donald Schön, writing about professional practice four decades later, gave the move its sharper name — *problem setting*, distinct from *problem solving*, and according to him the under-recognised half of skilled work. As I see it, conceptualization is problem setting, scene setting, and context setting all at once, and it is the half of the collaboration that the agent has the least access to.

The agent will faithfully solve whatever it is set to, often exhaustively. That is, to no small degree, much of what makes these tools so useful. It will, however, solve a wrongly-framed problem with the same care it brings to a rightly-framed one, and the costs of the former scale with capability. A correctly-solved wrong problem is a familiar failure mode at any level of skill; what is new is that the executor is fast, cheap, and tireless, which means the cost of the framing error compounds quickly and visibly. The instinctual urge is to continue throwing good tokens after bad, but that's merely an old logical fallacy in a new hat.

There is a useful family of examples here. A request for a search bar is sometimes a search bar problem and sometimes an information-architecture problem. A request for more reporting is sometimes a reporting problem and sometimes a measurement-design problem. A request for more tests is sometimes a coverage problem and sometimes a code-shape problem resisting verification. In each pair, a competent agent can satisfy the surface request to the letter, and ship a worse system in doing so. The human work is the recognition that the second framing bears examination, and the small act of will that says *let us step back, identify the original problem, and solve that one instead*.

What gives the human the standing to make that call, I would propose, is the same institutional and subject-matter knowledge I tried to characterise earlier. Reframing depends on having been present for the meetings, the failed attempts, the half-finished migrations, the shifts in stakeholder priority that no document quite captures, and the ability to conceive of what the stakeholder actually means as opposed to what they say. Without that ambient ground and experience, the most you can do is solve the problem as stated. With it, you can revise the statement, peel back the layers and get to the truth of the matter. The agent can be supplied with more context, but it cannot revise the framing of the problem it has been handed; adding context to a model pointed at the wrong problem only buys you a better-justified wrong answer.

This also reframes (forgive the word) what curation is asked to do. If conceptualization is upstream of curation, then a sharper conceptualization rewrites what is worth curating in the first place — entire bodies of carefully maintained material can become irrelevant under a re-framing, or suddenly relevant under another. The two disciplines compound. Better framing tells curation what to keep

close; well-tended curation surfaces the material that enables the next reframe.

A small concession before closing. There is a sense in which constraints already sit upstream of conceptualization, too — the choice to write a novel without the letter *e* is, in a way, a constraint that defines a problem. The Oulipo as a project has always been a little ambiguous on this point, and I think the ambiguity is honest: framing and constraining are mutually shaping moves, and pretending otherwise oversimplifies the work. What I would say is that the act of *choosing the frame at all* — of deciding that the project on the table is a novel without an *e*, rather than a novel about something else, or not a novel — is the conceptualization, and the rest of the discipline is what follows.

The distinctively human contribution in agentic work is the question itself. Constraints, Context, and Curation each presume a problem statement; Conceptualization is the move that produces one. And because the cost of a wrongly-framed problem rises with the capability of the executor, the value of getting the framing right concentrates as agents improve, rather than diminishing. That concentration, I believe, is the spine of the argument I have been building.

What lies on the other side of conceptualization is the *generative* work that happens once the framing is in place — the surface where the human contribution is most often invoked, and least precisely understood.

Creativity

The conversation about AI's limits has an inexorable destination. Whether you start it in a meeting room, a podcast interview, or a hallway after someone has demoed something agentic and impressive, sooner or later someone offers the closing line: *yes, but the AI can't be creative.* It is a reasonable thing to say, and as a working hypothesis I do not think it is wrong. But for an argument that has gone five Cs to get here, *creativity* is too coarse a term to leave unexamined.

There are at least three things people mean by it, and the distinctions matter. The cleanest typology I know — and the one that has held up best across cognitive science, AI research, and creativity studies — is Margaret Boden's, from *The Creative Mind: Myths and Mechanisms* (1990; 2nd ed., Routledge, 2004). She names three kinds of creativity that differ in kind, not in degree.

The first is **combinatorial**: novel combinations of familiar ideas. The malaphor that startles out a chuckle, the grasping analogy that lands sideways and is suddenly useful, the unexpected pairing of two flavours. Most everyday creative work is combinatorial — jokes, brainstorming, much of writing. The second is **exploratory**: the disciplined exploration of an existing conceptual space. New theorems within an existing axiomatic system, new poems within an established poetics, new chess positions within the rules. The bulk of what professionals call creative work is exploratory — novelty without rule-breaking. The third is **transformational**: changing the conceptual space itself. Cubism. Non-Euclidean geometry. The symphony as a form. The rarest, highest-stakes mode, and the one that tends to be the headline when the word *creativity* is invoked in a culture-of-genius register.

Once those three categories are identified, the question of what generative AI is or is not doing becomes much more answerable, because the answer is different for each.

Combinatorial creativity is, more or less, what a language model is for. Recombining material from a training distribution into novel surface configurations is essentially what the architecture does. The output may be uneven and the model may miss the *good* combination, but

the operation itself is well-suited to the tool. The human contribution at the combinatorial level is mostly upstream — setting the combination space well and selecting from the output — and that, properly considered, is a Constraints and Curation problem more than a Creativity one. The work is real, but the label is insufficient.

Exploratory creativity is where models attempt the work and tend toward the median. Exploring a conceptual space well requires reaching its edges, not its centre — and a model trained on averaged data has been pulled, by definition, toward the centroid of that data. Left to itself, a model will produce competent, plausible work that lives near the middle of the distribution. The human contribution at this level is not generation but direction: pushing the model toward the edges of the space, recognising when an output has actually reached one of those edges, as opposed to having merely surfaced a less common centre, and refusing outputs that look exploratory but are combinatorial in disguise, or pure hallucination that is closer to "nonsense in an answer-suit" than anything meaningful. This is taste expressed as steering, and it is not separable from the Constraints, Context, and Curation that have come before. They are how you get the model out to the edge in the first place.

Transformational creativity is still mostly, or entirely, human. Changing the conceptual space requires understanding its present shape well enough to conceive of how it could be otherwise — and that is contextual and historically situated in a way models trained on the already-said struggle to reach. A model can mimic transformational outputs after the fact, since it has been trained on what came after the transformation; it cannot straightforwardly initiate the move. The Cubist break and the move from Euclid to Riemann are both, in their own way, departures grounded in deep familiarity with the system they are leaving. It's not new information that Picasso was an artist with an extensive formal background and profound technical skill before he decided to break the mould. That depth is the institutional and subject-matter context I have been describing for four sections. It is what the human possesses that the model cannot, and it is at this threshold that the impact is most concentrated.

Standing back from the three, the structural asymmetry under all of them is the same. A model's reach is bounded by the training distribution: wide, mapped, predictably densest at the centre. The human's reach is bounded by lived experience: narrower in volume, but uncatalogued, historically situated, and capable of pulling in the parallel from another domain at the moment the model would have produced its median guess. That asymmetry is what makes the human contribution distinctive across all three Boden types. At the combinatorial level we supply selection from the model's fluent output. At the exploratory level we supply the steering toward the space's edges. At the transformational level we supply the contextual grounding that allows the conceptual space itself to be re-imagined.

The other four Cs are what produce that grounding. Constraints set the space within which exploration happens. Context is what makes transformational creativity possible — the deep, situated understanding of the present paradigm and what it costs. Curation is the discipline of selecting from combinatorial output where the model is fluent but not selective. Conceptualization is the human framing-move that opens exploratory or transformational space in the first place. Creativity, in this picture, is the surface the other four feed into; without them, we're grasping at straws or rolling dice, and the results are as random as a madlibs game.

Creativity is the human's job is too coarse a slogan to be useful. Boden's typology gives the slogan its honest reading: combinatorial work is increasingly delegable; exploratory work demands edge-seeking direction the model cannot supply by itself; transformational work depends on contextual grounding

that, for the moment, only the human can carry.

The advantage the human contribution holds across the three is not a separate creative spark sitting next to Constraints, Context, Curation, and Conceptualization. It is the same advantage, expressed at the surface where it matters most. The other four Cs are the sources; Creativity is where they show their work.

Conclusion

And so we conclude the walk through my map of the five Cs. Let me close by summarizing what they share.

Constraints narrow the neighbourhood of outputs we will accept. Context shapes the inputs the model is working from. Curation decides which slice of available context earns the prompt this time. Conceptualization decides what the work is for. Creativity is the surface the other four feed into. Each of the five is its own discipline, and each pays for itself even in isolation; that's why I have given them their own sections. But the five are not completely separable in practice, and the argument I have been building is finally about what they share, not what distinguishes them.

The shared structure is an asymmetry, and it is the same asymmetry from one C to the next. The model has the training distribution: a collection of data assembled into shapes systems can swiftly consume, carefully constructed, weighted and tuned to provide the best possible responses to the largest set of people. The human has lived experience: narrower in volume, less completely catalogued, specific in one strain of experience, and capable of associations and inspirations the model is highly unlikely to supply. At every category — every "C" — the human contribution is made of what the training distribution cannot reach. Constraints narrow toward what the model could not reasonably weigh on its own. Context supplies what the model was not trained on. Curation is the judgement of which of those un-trained inputs is worth paying for *now*. Conceptualization is the framing the model cannot revise from inside the prompt it has been handed. Creativity is where all of those upstream contributions meet the generative surface.

This is, I would propose, the most honest reading of the claim that in the last of those, the human surpasses the tool. Not that the work is sacred or that the human's role is somehow elevated, but that the work the human does best is the work the tool is not designed to do — the work whose inputs the tool cannot supply itself. The five Cs hold up as a "map" because they trace exactly the categories of effort where that surpassing happens — and, as a consequence, where the tool's acceleration of the human's work concentrates.

What does any of this change for the practitioner? Mostly the locus of their attention. If the load-bearing work happens upstream of the model, the practitioner's attention moves there with it. Time spent shaping a prompt is time spent shaping a constraint, a context, a curation, a conceptualization — and, downstream, a creative surface. The prompt is where the work shows up, but the prompt is not where the work is. The work is in the gathering and the trimming and the framing and the choosing-of-the-question that produced the prompt. It's in what artefacts you provide as reference, and what demands you instate for the output. Skip those, and the agent's fluency will still produce something — likely something demonstrable, possibly even something useful — but the work that was meant to be done will not have been done, and the acceleration that was on offer will quietly not have arrived.

These tools will keep improving, and I would not bet against any particular corner of this argument. The training distributions will widen. The retrieval architectures will sharpen. The cost of context

will continue to fall. What I would bet on is the asymmetry itself. As long as there is institutional knowledge no one has written down, situated judgement under pressure, frames that cannot be revised, and leaps a model cannot make, attending to these five categories will be where the human contribution shows its strength — and, as a consequence, where the tool's potential as an accelerant is most fully realised. The locus has shifted before — from punch cards to IDEs to prompts — and will shift again. At each shift the practice of attending stays where it always sits: with the person who knows what the work is for.

Constraints. Context. Curation. Conceptualization. Creativity. A working programmer who focuses on all five, on each piece of real work, will produce better work than the one who does not — and faster work as a consequence, almost as an afterthought. The five Cs are my map of that territory. But, as always, the real work is to walk it.