Where the Artefacts Live

1. The framing that misses what changed

The dominant framing of writing in AI-assisted engineering is that documentation has become more important. AI tooling needs context to work well, so teams that previously got away with thin documentation now have to write more. The framing is not wrong as far as it goes. It misses what has actually changed.

What has actually changed is not that documentation matters more. What has changed is that writing in engineering now serves several distinct functions that used to be combined or absent. The pre-AI taxonomy of engineering writing — design documents, code comments, commit messages, runbooks, API documentation — assumed each kind of writing had a clear human consumer and a clear function, and that the engineer producing it could usually identify both. This is no longer true. New categories of artefact have appeared without obvious names or homes; existing categories have acquired second functions that pull in different directions from their original ones. Treating all of this as "documentation that has become more important" produces practices that work for one function at the expense of another, because the functions themselves have not been clearly distinguished.

This piece argues that writing in AI-assisted engineering now serves at least four distinct functions, that these functions operate across multiple organisational levels, and that the practical question of where each artefact should live cannot be answered until both its function and its level have been named.

2. What writing used to do

Before AI, writing in engineering was overwhelmingly consumed by humans and overwhelmingly optional. Design documents, code comments, commit messages, runbooks — each had a clear human consumer and could be skipped if the team was willing to accept the consequences. The code was the artefact that mattered; writing was supplementary. Both of these features have changed. Writing in AI-assisted engineering is no longer optional in the same way, and it is no longer consumed only by humans. These changes are what produce the proliferation of functions the rest of this piece examines.

3. The functions that writing now serves

Four functions are worth distinguishing.

Context. Some writing exists to make AI tooling work at all. Project READMEs, conventions documents, architectural overviews, agent role definitions, skill instructions, system prompts. The consumer is the AI tooling itself. The function is to shape what the AI does — what conventions it

follows, what assumptions it makes, what it produces. If context writing is wrong or stale, the AI produces output that is fluent but subtly disconnected from the system it is supposed to be operating within. The failure mode is silent: AI tooling does not signal that its context is inadequate.

The new category here is *deliberately authored instructions that shape what the AI produces*, which had no clear pre-AI equivalent. The closest historical analogue is the style guide, but style guides were aspirational human references; context artefacts are operational AI inputs. The difference matters for how they are written, how often they are updated, and what happens when they drift.

Handoff. Some writing exists to carry state between activities during a single piece of work. Bug reports passed from an investigator to a planner, change plans passed from a planner to an implementor, descriptions of changes passed to reviewers. The consumer is the next activity in the workflow. The function is to compress the state of the work so far into a form that can serve as input to what comes next. The durability requirement is short: handoff artefacts matter while the work is in flight and lose most of their value once the work is complete.

Handoff writing is genuinely new. A single engineer carrying work from investigation to implementation did not need to write a handoff document between activities they were performing themselves; the state lived in their head. When AI agents perform discrete activities and require explicit context to do the next step, handoff documents become necessary. They are the externalised form of what used to live in a single engineer's working memory.

Evidence. Some writing exists to document what was done after the fact. Pull request descriptions, commit messages, ticket comments, change logs. The consumer is varied — reviewers, operators investigating incidents, managers, future engineers trying to understand why a change was made. The function is to leave a trail. Evidence writing existed before AI, but its role has changed: pre-AI, the code itself was partly the evidence, because the act of writing it demonstrated understanding. AI breaks this linkage. Code that the engineer barely looked at can be shipped, which means evidence has to be produced separately and deliberately, because the code can no longer be assumed to carry it.

Reference. Some writing exists as durable documentation of how a system works. Architectural decision records, runbooks, design documents that survive the project that produced them. Reference writing has not disappeared with AI, but its role has acquired a second dimension. Reference artefacts are now consumed not only by humans seeking to understand a system but also by AI tooling seeking context for what it produces. Writing optimised for human reference often makes assumptions about background knowledge and assumes the reader will skim and re-read; writing optimised for AI context needs to be more explicit, more structured, and more current. Teams that have not yet noticed this dual role tend to write reference documents that serve one consumer well and the other poorly.

4. Why the four functions matter

The dominant impulse when faced with the proliferation of writing in AI-assisted engineering is to treat it all as "documentation" and try to make it better — more thorough, more current, more structured. This impulse is not wrong, but the four functions have different requirements that pull in different directions.

Context writing needs currency above all else. A stale context artefact actively misleads AI tooling. Handoff writing needs precision but does not need durability — a handoff document that perfectly captures the state of work in flight has done its job even if it becomes useless the moment the work completes. Evidence writing needs durability and searchability but its real-time precision requirements are weaker. Reference writing has to balance human and AI consumption, which often pulls in opposing directions.

The practical consequence of confusing these functions is that teams optimise for the wrong properties. A team that treats all writing as evidence writing produces durable but stale context artefacts. A team that treats all writing as context writing produces precise but unreadable evidence trails. A team that treats handoff documents as reference documents over-invests in artefacts that will lose their value within days. The optimisation is local; the misalignment is global.

5. The second dimension: organisational level

The four functions are one dimension along which artefacts vary. There is a second dimension worth naming: the organisational level at which an artefact applies. The same function can manifest at multiple levels, and the level affects practical decisions about authorship, update cadence, and authority.

Consider context artefacts. A `CLAUDE.md` in an engineer's home directory is a personal context artefact, applied by AI tooling when that specific engineer is working. A project README is a team-level context artefact, shared by everyone working on that codebase. A coding-standards document might be an organisation-level context artefact, applied across many teams. All three are context artefacts by function, but they live at different levels and serve different audiences. A personal artefact can be edited freely by its owner; a team-level artefact requires team agreement to change; an organisation-level artefact carries consequences across many teams when it shifts.

Not all functions span all levels. Handoff artefacts are almost always workstream-specific — they belong to the piece of work they support, not to any persistent organisational unit. Reference artefacts tend to be team-level or organisation-level, rarely personal. Evidence artefacts vary depending on what is being evidenced. Context artefacts span the full range, which is part of what makes them the most operationally complex of the four functions.

The practical implication is that the question of where an artefact lives has two answers, not one. *What kind of system* — repository, ticket tracker, personal file — is the first answer. *At what organisational level* — personal, team, organisation — is the second. Both have to be right for the artefact to do its work.

6. Where the artefacts should live

Once the four functions and their levels are distinguished, the question of where each artefact belongs becomes more tractable, though it does not resolve into clean answers.

Context artefacts probably belong with the code at every level they apply to. Their currency requirements demand version control, and they need to be updated when the conventions they describe shift. Personal context artefacts can live in an engineer's home directory or personal configuration; team-level context belongs in the team's repository; organisation-level context belongs in shared repositories or platforms that all teams consume. Putting context artefacts in places that are not part of normal engineering workflow tends to produce drift, which produces silent failure in AI tooling.

Handoff artefacts probably belong with the work they support, not with the code they describe. Their durability is short and tied to specific tickets or workstreams. The natural home is wherever a team already tracks active work — issue trackers, ticket systems, the running record of what is currently being done.

Evidence artefacts probably belong in some combination of version control history and ticket systems. Pull request descriptions and commit messages live with the code; broader narrative descriptions of changes live with the tickets that drove them.

Reference artefacts have the most complicated home, because of the dual consumption problem. The best practice is probably to maintain reference writing in a form that can serve both consumers — usually structured markdown in version control with summaries and indexes that make it navigable for humans.

These are tentative answers. The point is not that they are correct in detail. The point is that they are different for different functions and different levels, and a single home is not the right answer for all of them. The impulse to put all written artefacts in one place — the repository, or the ticket system, or a separate documentation portal — will produce suboptimal homes for at least some of the functions and levels writing now serves.

7. Closing

The dominant framing treats writing in AI-assisted engineering as documentation that has become more important. The framing is not wrong but it is incomplete. What has changed is not that documentation matters more. What has changed is that writing in engineering now serves several distinct functions, with different consumers, different durability requirements, and different failure modes, and these functions operate across multiple organisational levels. These distinctions have not yet been clearly drawn in how teams talk about and organise their written artefacts.

This piece is not a claim that the four functions named here are the only ones, or the right cut, or that they will remain stable as practices around AI-assisted engineering develop. It is a claim that treating writing as one undifferentiated category called documentation will produce practices that

work for one function at the expense of others, and that the impulse to revert to familiar homes for new artefacts may be sensible in some cases and a default in others. Distinguishing which is which requires being clearer about what each artefact is actually doing, and at what level it applies.

Engineering teams are producing more written artefacts than they used to, in more distinct varieties, and the question of where each variety belongs is not yet settled — partly because the varieties themselves are not yet clearly distinguished. Asking what each artefact is actually doing, and at what level, before deciding where it lives, is the work that has to be done.