

Mission Tools Must Carry Their Intent

By: [Matt Hooper](#), Principal Engineer and Site Lead

A maintenance lead spends an afternoon with AI building a tool for the handoff between shifts. It pulls open work orders, checks the latest notes, and flags the issues that need attention. By the end of the day the incoming shift gets a clear account of unresolved work, and the lead understands their own handoff better than they did that morning.

That tool is operator-built AI: software the people who own the work build for themselves, without waiting on a development queue. It is also a ready starting point for the next shop that needs a handoff. The next shop starts where the first lead finished, builds on what they learned, then shapes the tool to its own equipment, records, and schedule. For that to work, the tool has to carry its intent, meaning the purpose it serves, the limits it must respect, and the reasons behind its local choices. A tool that carries only its behavior is tribal knowledge in software form.

AI Creates Tribal Knowledge Faster Than Teams Can Share It

The shift sees the report, while the reasoning behind it stays with the person who built it.

Tribal knowledge is expertise that lives with one person instead of the team. The maintenance lead's tool is full of it. They know which records arrive late, which incomplete notes need a conversation, and when an unresolved issue needs a supervisor. Each revision during that afternoon encoded something they know about the shop. Their colleagues see a working report. Few of them see why it works.

Redshift is the name for the growing distance between what the builder knows and what the team can see. In astronomy, cosmological redshift is the stretching of light as space expands, and it is how astronomers measure galaxies moving apart. With AI, one person's understanding moves forward faster than the team can absorb what changed and why. Individual productivity rises while the team's ability to build on that progress falls behind.

Anthropic found early signs of this inside its own engineering teams. In [a December 2025 study of how AI changed work at the company](#), employees described Claude as "the first stop for questions that once went to colleagues." One senior engineer named the cost: "It's been sad that more junior people don't come to me with questions as often." [1] The questions that used to spread expertise across a team now go to a model, and the answers stay in one conversation.

Turnover makes the cost visible. In a military maintenance unit, the lead who built the tool will move on through a permanent change of station, a deployment, or a reassignment. The tool keeps running. The next lead inherits an escalation rule that flags any issue still open after one shift, and nothing in the tool says whether that rule enforces a safety requirement or reflects the previous shift's staffing.

The new lead has to answer that question [during a shift change, when people need to act](#). A supervisor tries to reach someone who has already left. The tool saved effort producing the report, and [the unit still depends on its original builder to explain how to use it](#).

That dependence is the warning. The fix starts during the build, while the decisions are fresh. When the lead explains why an issue needs attention or why a threshold changed, that explanation stays attached to the tool. The replacement then has a place to start when the same question comes up, and so does every shop that wants the tool next.

The record does not need to be long. For the escalation rule, it is a few lines: the rule flags issues open after one shift; it exists because a supervisor is on hand during the overlap and records are current by then; the lead set it after an unresolved hydraulic leak sat for two shifts without anyone above the line seeing it. A replacement reading those lines knows what the rule protects and what conditions it depends on. That is enough to decide during a shift change without a phone call.

Copied Tools Carry Assumptions That No Longer Fit

A second shop can copy the tool unchanged or rebuild it from scratch, and both paths lose the first lead's expertise.

Copied tools look authoritative because they arrive as working software. A second maintenance shop wants the handoff tool. Its equipment differs, its records arrive later, and its shifts overlap on a different schedule. Both shops need continuity between shifts. Neither needs an identical report or identical escalation timing.

The first path is to copy the tool unchanged. In the first shop, records are current by shift change and a supervisor is available during the overlap, so flagging any issue still open after one shift makes sense. In the second shop, routine updates post hours later, and the same rule flags work that is finished but not yet recorded. The incoming shift has to sort confirmed problems from paperwork lag. Without the first shop's reasoning, nobody can tell whether changing the threshold removes a safeguard or fixes a scheduling mismatch. So the rule stays, and the team goes back to the records to check every flag. Copying moved the mechanism and left the expertise behind.

The second path is to rebuild from scratch. The second lead opens a new conversation with AI, reconnects the records, and rediscovers which notes need a human to interpret them. Generating a new tool takes an afternoon. Recovering the judgment behind the first one takes much longer, and some of it never comes back.

Software engineers learned this long before AI. In his [2000 essay on Netscape's decision to rewrite its browser](#), Joel Spolsky observed: "It's harder to read code than to write it." Old code looks messy because it carries fixes from real use, and each fix is a lesson whose purpose is easy to miss. Starting over throws those lessons away. [2] AI makes a rewrite faster to produce and does nothing to bring the lessons back.

Neither path gives the second shop what it needs: a starting point it can understand, then make its own.

Commander's Intent Lets the Approach Change

The second lead needs to know which rules protect the mission and which only reflect how the first shop runs.

Commander's intent is how the military already solves this problem. It [communicates purpose and the desired end state](#), so subordinates can choose how to act within prescribed limits when circumstances change, without reconstructing every planning discussion. [3] Mission command depends on it: shared understanding of what matters, with discretion over how to achieve it.

Useful context has to carry forward so no team starts from [an empty chat box](#). Commander's intent defines what that context has to contain. For a tool moving between shops, it has three parts.

The purpose is what both shops are trying to achieve. For the handoff tool, that is continuity between shifts: incoming maintainers understand unresolved work, why it matters, and who needs to act.

The limits are the authority boundaries and safeguards that hold in any shop. The handoff tool summarizes and flags work. It does not release equipment for service or replace the authorized maintainer's judgment. Required reviews and access restrictions stay visible to the receiving team, and each one names its owner, so a local edit cannot quietly change who decides.

The local choices are how the first shop implemented the purpose: its record connections, report order, and escalation timing. Each choice carries the assumption behind it. The one-shift rule assumes records are current at shift change and a supervisor is on hand during the overlap. Once the second lead can see that assumption, they can see exactly where their shop differs.

Evidence travels with the tool as well: the conditions under which the tool was last assessed and the questions still open. If the first shop only ever ran the tool with current records, the second lead has a specific assumption to test before trusting the flags. That turns an inherited rule into a known starting point.

Routine users never need the whole history. An incoming maintainer needs a short list of urgent issues. A maintainer chasing an unusual result needs a path to the reasoning behind one flag. Neither needs every exchange from the original build, and the record is organized so each finds what they need through the work they are already doing.

Picture the structure as a barbell. One end defines the goal, the conditions that count as success, and the limits. The other end assesses whether the work achieved that purpose. The implementation between them is free to change, and the two ends keep that change tied to the purpose.

For the second shop, adaptation starts from those ends and its own conditions. Its lead identifies when records become trustworthy, which issues require immediate attention, and where the inherited timing no longer fits. They adjust the connections, timing, or layout and keep the purpose and limits intact. The result is a tool the second lead understands and owns, built on what the first lead learned.

Leaders set the boundaries of that discretion. Changing the report order is a local choice. Changing who approves a maintenance action is not, and the receiving lead needs to know the difference before editing anything. The explanation also stays attached to the version in use. When the second shop changes a threshold, the new reason and the conditions behind it go with the change. Otherwise today's informed adaptation becomes tomorrow's unexplained rule, and redshift returns inside the tool that was supposed to close it.

Some teams want a different outcome altogether. A handoff tool is a reasonable starting point for long-range maintenance planning, but the two jobs have different success conditions. The team states the new purpose and reassesses the inherited approach against it. Evidence that the tool works for a shift handoff says nothing yet about planning.

Success Is Measured in the Handoff

A complete report can still leave the incoming shift missing the issue that matters most.

Success for the handoff tool is what the incoming shift understands and does. A finished report is only the first piece of evidence. [The Army's FM 5-0](#) ties assessment to objectives and end-state conditions and keeps the plan's logic visible, so assumptions

and the approach get reconsidered as events unfold. [4] That is the other end of the barbell.

Assessment starts with checks on the tool itself: source records appear accurately, required reviews stay intact, and missing information is flagged. Then it moves to the handoff. Incoming maintainers identify the unresolved issues that matter, name who owns the next action, and recognize when the records are too incomplete for a judgment. In the second shop, that includes telling a confirmed open issue from a flag waiting on a record update.

When the incoming shift misses an important issue more than once, the cause matters more than the miss. Three causes look identical from the shift's side: a stale source, a poor summary, or a label the receiving team reads differently. Each calls for a different fix, and each fix is recorded with its reason so the next team inherits the lesson along with the revision.

The transfer itself gets tested with the people inheriting the work. The second shop receives the tool and the explanation of its purpose, limits, and local choices. Its lead inspects their own records and adapts the approach. Two results show whether the transfer worked: whether they catch the escalation assumption that doesn't fit, and whether the handoff runs without a call to the original builder. If they need the first lead to say which rules are requirements, the explanation is incomplete. If they understand the reasoning and the handoff still fails, the assumptions or the approach need another look.

Over time, both shops' changes and results feed one shared account of the work. Missing context that shows up in both shops improves the common explanation. Differences that come from operating conditions stay local choices. That separation keeps one shop's successful change from becoming an unexplained requirement for every shop that follows.

Conversations with the original lead still matter, especially the repeated ones. When three incoming leads ask the same question about the same flag, the answer belongs in the record, and each transfer leaves a better starting point than the last.

Every Build Must Leave the Next Team a Better Starting Point

Leaders decide whether operator-built AI becomes something the next team can build on when they define what a finished build is.

A finished build includes the transfer. The originating team records the purpose, important decisions, limits, and evidence as the work develops. The receiving team owns its adaptations and its assessment. A build is judged by what the next team understands and accomplishes, including after the original builder moves on.

AI is useful for keeping that record, with one hard rule: an explanation generated afterward stays distinguishable from the reason someone actually made a decision. When the original rationale is unknown, the record says unknown. A plausible account of why the first shop chose a threshold is a hypothesis to test, not a fact to inherit.

Legion Intelligence designed Legion Mission Builder so the limits stay fixed while operators change the rest. Teams start from the Legion Pack closest to the work, or describe the work in plain language, and shape it into their own application. Mission Builder proposes the steps, and the person building confirms each one before anything runs. Applications are composed from parts that carry permissions, classification handling, and audit, and [administrators keep authority over permissions and the approvals each step requires](#). A builder composes only from the parts they are cleared for and cannot grant themselves new access.

That design does not yet prove the transfer. The test is the one described above: a second team receives a tool, understands which choices matter, adapts it to its own conditions, and runs a successful handoff without calling the person who built it. That evidence comes from use, shop by shop.

The next maintenance lead who opens that handoff tool will never meet the person who built it. If the reasons travel with the tool, they will not need to.

See how [Legion Mission Builder](#) shapes a [Legion Pack](#) to your unit.

References

1. Anthropic. [How AI Is Transforming Work at Anthropic](#). December 2, 2025.
2. Joel Spolsky. [Things You Should Never Do, Part I](#). Joel on Software, April 6, 2000.
3. Gen. Gustave Perna. [Mission command requires sharp commander's intent](#). Army Sustainment, January–March 2019; republished November 4, 2019.
4. Headquarters, [Department of the Army. FM 5-0 Planning and Orders Production](#). November 4, 2024, incorporating Change 1, August 2025. Paragraph 4-99 and chapter 8.

Continue the series

Follow Legion Intelligence on LinkedIn: [linkedin.com/company/legion-intel](https://www.linkedin.com/company/legion-intel)

Explore all Command Papers: legionintel.com/command-papers

Ready to talk: legionintel.com/contact