TEMPLATE · FILL-IN REFERENCE

Definition of Verified

Ed Schaefer · Version 1.0 · August 2026 · edward-schaefer.com/templates/definition-of-verified · Free to share

Every team putting agents into real work eventually has to answer one question: verified by whom, and against what standard. Most teams answer it for the first time only after something ships that shouldn't have. This template gets the answer written down before that happens, not after.

1. What "verified" means here

Starting definition, marked as a default: replace it only if your team has a better one.

Verified means someone other than the one who produced the work checked it against a standard, and left evidence that they did. Not that it looks right, and not that the agent says it passed.

Trust in the output is not the same thing as verification of it — reliability comes from checking, not model magic. The shorter version: trust but verify.

2. What counts as verified, by work type

This maps onto the five-tier blast-radius scale from Ed's published essay, Human-in-the-Loop Is a Coaching Problem, summarized here so this table works without reading it first:

- **Tier 1 — just do it.** Small, low-risk, reversible. No ask.
- **Tier 2 — do it, tell me, let me green-light.** A little input, then proceed.
- **Tier 3 — a quick exchange first.** A small decision, or a manual step someone has to take.
- **Tier 4 — a real conversation.** Evaluate options, maybe a research spike, maybe someone does something by hand.
- **Tier 5 — fully human.** Standing up infrastructure, anything irreversible, anything someone has to own.

Autonomy granted = earned trust × blast radius: high trust and low blast radius runs on its own; low trust or high blast radius keeps a person engaged, regardless of tier. Blast radius alone sets your default review depth here:

Rule of thumb for placing a work type on the scale: ask what breaks if this is wrong, and whether it can be undone. Small and reversible sits low; anything real users depend on, or that can't be undone, sits high, however simple the task looked going in.

Blast radius	Tier	Default review depth
Low	1	Automated checks that fail loudly. No person required by default.
Low	2	Automated checks that fail loudly, plus the green-light the tier already requires. The green-light authorizes the work to proceed; it isn't verification of the result. Verifying the result stays with the automated checks.
Middle	3	Sampled human review.
High / irreversible	4, 5	Full human verification. Stop-the-line authority is strongest here.

These tiers set prevention — how much checking a piece of work needs before it ships. That's a different question from how hard to react if something gets through anyway, which is the Agent Incident Runbook's severity ladder (section 2 there).

Assign your own work types to a tier, then fill in the rest. One worked example is provided as the default. Replace it, don't keep it:

Work type	Blast-radius tier	Who verifies (not the author)	What "verified" requires	Evidence produced
Default example: Code that reaches real users	4	A person who did not write it	Runs it, or reads the change line by line — not just the summary	A dated review record naming what was checked

A row with nothing in the last two columns means the work was assumed to be fine, not verified.

3. Who can verify what

The authority to approve work follows the table above — at the low tiers, that can be an automated check instead of a person. The authority to *stop* work is different: it belongs to whoever is closest to the defect, not only to whichever role has the title.

Toyota's Andon cord is the model: any assembly-line worker could stop the entire line the moment they spotted a defect, without asking permission first. This template asks your team to run the same rule.

The stop-the-line rule: anyone on this team, regardless of role, seniority, or whether they touched the work, can stop the flow the moment they spot a defect.

- Can stop the line: _____ (default: anyone on this team, per the rule above — this line is for naming scope, like which systems it covers, not for narrowing who's allowed)

- The line stays stopped until: _____ (default: the defect is addressed, not until someone senior says otherwise)

4. What evidence verification produces

Default: a dated record, kept somewhere the team can find it again, of what was checked and who checked it.

"I looked at it and it's fine" doesn't count as evidence, and neither does an agent reporting that its own output passed. Never rubber-stamp. Evidence answers three things: what was checked, against what standard, and what happened when it was. At the low tiers above, an automated check that fails loudly clears this bar on its own. Once that record exists, nobody has to hold the explanation in their own head anymore.

One row is standing, regardless of work type: for anything with system or data access, evidence has to say what the agent's credentials could reach, not only what it was instructed to do. Add it every time the work touches a real system.

Fill in for your team: -Where evidence is kept: _____

5. What happens when verification is skipped

Skipping the check doesn't remove the defect. It moves the cost later, where it's larger. The source post's own point about manufacturing applies without modification here: defects become exponentially more expensive to fix the longer they go undetected.

The source post quotes Sidney Dekker on exactly this point — a line that reaches Ed's shelf through The DevOps Handbook's citation of Dekker, not a Dekker book directly: "Meeting your schedule and budget today is no guarantee for meeting your schedule and budget tomorrow. Sacrificing resilience for short-term efficiency is dangerous." Skipping a check to hit today's deadline borrows against tomorrow's.

Fill in for your team: -What actually happens here when verification gets skipped: ___ **- Who absorbs the cost: ___** -When it shows up: _____ (default: later, and bigger than it would have been at the check)

This is a fit question, not a recipe

The shape above is a strong default. Where your team actually sits against it, and what that's costing, is what the Engineering Ways of Working Diagnostic is built to find.