



Ontollo for Construction. *Turning field reality into operational decisions.*

The moment

Every general contractor and subcontractor already runs good software: PlanGrid, Procore, Epicor, InEight, Dynamics, plus whatever point tools a project team picked up along the way. That's not the problem.

It comes down to a tale of two economies. AI has mastered *knowledge work* (documents, emails, best practices) because that work is static language a copilot can read. It has not yet mastered *operational work* (sensors, machines, people, weather, the job site that changes by the hour). Operational work is dynamic and locationally precise — the location and timing of the work *are* the work. Ontollo was built for that second economy.

Every operation has a **system of record** (ERP, CRM, project management systems) that captures what already happened. Almost none has a **system of reality**: a live, connected model of what is happening right now across every asset, signal, and person. **Ontollo is that layer.** We marry the **operational signals** from your crews, equipment, and field activity with the **business drivers** (timeline, budget, cost codes, contract terms) and the **organizational knowledge** your best people carry in their heads. Fused into one living model, that context lets AI reason over your operation and return concrete moves: decisions and tasks routed to the right person, agent, or device—not another report.



How the problem plays out

When a pour happens at a specific site on a specific day, precision matters. Equipment at Site A that needs to move to Site B ripples to three other projects. A crew's location changes the sequence of the next three tasks. A precision mistake cascades: a pour spec becomes a rework, an unbilled hour becomes a margin leak, a missed connection becomes a schedule slip that compounds down the line.

The problem is that the systems tracking this capture reality only if skilled people stop doing their real job to feed them in. A superintendent who knows a pour cold still spends hours re-entering it, field by field, into a daily report. An installer who knows exactly how a builder's spec differs from the standard still has no system that catches the difference before it becomes a rework order.

Ontollo is built to close that gap: a governed layer that turns field reality, business drivers, and the judgment of your best people into decisions your team can act on, without adding another system for the field to feed.

Reason over reality, not the record of it

The default move in operational AI is the copilot paradigm: take the AI that works for knowledge work and point it at operations. It fails for one reason. Knowledge work *is* language, so an assistant that reads and writes language is enough. Operational work is not language. It's physical things, in states, at locations, over time, in relationships. The report, the daily log, the signal feed *describe* the work. They are not the work itself.

Ontollo builds a live model of physical reality—what exists, where it is, how it relates, how it's changing—and reasons over that. We don't bolt a copilot onto the paperwork and call it operational AI. Three rules follow:

- **Capture from how people already work. Don't make reality a data-entry chore.** If the system's core ask of the field is "type in what happened," it has already lost to staleness. Stale operational data is almost worthless—it's faster to guess than to look it up. Precision comes from voice and inference, extracted from work people are already doing, not from a new burden laid on top of it.
- **Reason over the whole related model. Don't act on a slice.** An AI that sees only the schedule, or only the equipment, makes confident, coordinated mistakes. Equipment already committed to three projects looks available to a scheduling AI working in isolation. If the system can't see what depends on what, it doesn't get to act.
- **Measure success as toil removed. Don't ship another dashboard.** More visibility that costs the field more effort is a failure, even when the screen looks impressive. The win is a compliance box that checks itself and a billable hour that surfaces itself, not another thing to go look at.

Why this matters at construction's margins

Construction margins are notoriously thin, often mid-single digits net. McKinsey estimates that better use of project data alone can lift industry productivity by roughly 14–15 percent, with AI capable of adding meaningfully more once it has real operational context to work with.

On a business running a 3–5 percent margin, that's not a rounding error. It's the difference between a job that makes money and one that doesn't. Small leaks (an unbilled hour, a missed spec, a schedule slip nobody connected) sink the boat at these margins.

Proof: two construction stories

A concrete subcontractor's daily report, in under a minute instead of two hours.

A superintendent narrates the day's pour the way he already talks about it on site: quantity placed, crew, hours, equipment. Ontollo structures that into the daily report format the team already uses, and flags the cost code for confirmation. Because it understands the project's contract terms, it surfaces standby time that's billable back to a vendor and would otherwise go unclaimed. The system fits into PlanGrid and your cost-tracking tools; it doesn't replace them.

Faster data entry is only half the value. The same pour, once captured, also updates equipment utilization across every site, feeds next week's crew and material planning, and tightens the cost-to-complete view finance is staring at. One capture, a dozen downstream decisions, instead of one report nobody reads twice.



An interior finish installer, catching builder-spec variation before it becomes re-work.

Every production builder specs things differently: drill hole placement, hardware, cutout dimensions, all varying by floorplan and elevation. A cycle-time guarantee (installed today, verified tomorrow) leaves no room for a missed spec. Ontollo holds the builder-by-builder, floorplan-by-floorplan variation as structured knowledge, so the crew on site gets the right spec at the right moment instead of finding out after install.

Same underlying mechanism in both cases: field signals plus business context plus institutional knowledge, fused into one model, producing a decision or a flag instead of another form to fill out.

Why use a unified system of reality, not Palantir or piecemeal AI

Enterprise platforms like Palantir build unified systems of reality for operations. They work, but at a scale (cost, timeline, team, customization) that only makes sense for enterprises with a \$10M+ budget and a multi-year runway. For mid-market operators, Palantir is overkill, and the implementation tax eats all the gains.

Point AI solutions (scheduling AI, compliance AI, field-capture AI deployed separately) feel iterative and lower-risk. The problem: each operates in its own model of reality. When the scheduling AI moves equipment, it doesn't know that equipment is already committed to three other projects. When field-capture logs a slip, it doesn't feed cost-to-complete. Faster individual decisions coordinated worse than before is worse than no AI at all.

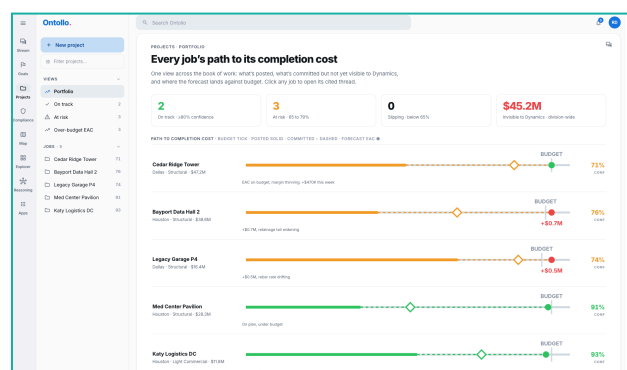
Ontollo is the unified system of reality, sized for the enterprise. Prebuilt, not custom-architected. Fast to stand up (weeks, not years). Governed, so AI reasons, while the sense of reality stays deterministic and cited. One model connecting the three things that matter: low-friction precision extraction (voice, not forms), a unified map of where things are and how they relate, and overhead removal while surfacing decisions. It's the middle ground Palantir's too big to reach and piecemeal AI can't actually achieve.

Operating principles we build on

- **There is no clean slate.** Your ERP, scheduling tool, telematics feeds, and point solutions all stay exactly where they are. Ontollo sits above them as the layer that connects what they already capture.
- **One faster workflow isn't the win. A living graph is.** Data captured once should inform a dozen downstream decisions (equipment, planning, billing, cost-to-complete), not close out a single report and go quiet.
- **Reality lives on the ground.** The field shouldn't have to become a data-entry function to make its own knowledge visible. Voice and existing workflows, not new forms.
- **Automate the audit trail, not the judgment call.** Contract clauses, standby billing, compliance sign-offs – exactly the painstaking, rules-based work that should be offloaded, so your people stay focused on decisions that need their judgment.
- **Superpower the crew, don't replace their expertise.** The reason a superintendent sequences a pour a certain way, or an installer catches a spec mismatch by feel, is exactly the knowledge worth capturing and reusing, not routing around.
- **Prove value in one use case, then expand from there.** We start with one bottleneck, fully verifiable and cited, and expand only as the system proves itself.

Architecture and trust facts

- **Grounded, not guessed.** Every input traces to a real report, sensor, or system reading. Any number driving a decision is computed, not inferred. If it can't be cited, it isn't shown.
- **Operator consent.** No action is taken without explicit approval. The system recommends; your people decide.
- **Canonical catalog.** About 50 prebuilt object types common to physical-asset businesses give the model of your operation a head start, so it comes together in days, not months.



- **Time to value.** Pilots stand up in about a week once data access is granted, and each integration we build makes the next similar one faster.
- **Model-agnostic.** Run on whichever AI engine fits your cost, performance, and data-location needs, so you are never locked to one vendor.
- **Data sovereignty.** Your data is never used to train the model, your know-how stays in a layer you control, and the platform can run on your own hardware for sensitive or remote sites.

How we work with you

Platform and partnership: we build it with you, in two low-risk phases. You keep everything we produce, regardless of what comes next.

PHASE 1 — DISCOVERY <i>30 days • no cost</i>	PHASE 2 — PILOT <i>60 days</i>
● Map the technology landscape and how data flows across your systems. ● A handful of interviews with the people actually running the work. ● Connect 2–3 real data sources into a working proof of concept. ● Identify the highest-value bottleneck and what the status quo is costing you.	● Build a working solution on the priority use case. ● Run it live, with real users, on a real project. ● Measure against success criteria agreed up front. ● A clear go / no-go decision on schedule.

Ontollo, Inc. - turning operational reality into business decisions. Reach out for a demonstration: www.ontollo.com.