SINGULARICS

White Paper

# Spec-Driven Development

The spec is a testable contract, written before the build. Speed of production raises the cost of ambiguity.

Alexander S. Petty
SINGULARICS | singularics.ai



First published August 2026.

singularics.ai

# Contents

## Abstract

*AI compresses the cost of building software toward zero, but it does nothing to the cost of building the wrong software. That cost rises, because faster production means more output to unwind when the direction is wrong. The spec is the instrument that pays down ambiguity before it compounds. This paper describes the practice of spec-driven development: what a spec contains, who writes it, how long it takes, and how a one-day cycle stays on course when the Decision Owner does not yet know what tomorrow holds. The central argument is simple. Speed of production raises the cost of ambiguity. The spec is how you keep that cost from compounding.*

## The Spec in the AI Era

In every engagement where we have introduced agent-driven delivery, the first week follows the same script. The fleet ships fast. The rework rate exceeds 40%. The team blames the agents. The actual cause, in every case, was that nobody wrote down what "right" meant before the build started.

The guiding principle has not changed. Work in small increments, sequence the highest value first, and get feedback quickly. What changed is the cost of an increment. When an agent fleet can produce a working implementation in hours, the constraint does not disappear. It moves to the slowest remaining human step: deciding what should be built, judging whether what came back is right, and supplying the domain knowledge needed to make either call.

That relocation changes one thing above all. When production was slow, a vague requirement caused modest rework. A developer would ask a clarifying question, wait half a day for the answer, and adjust. The ambiguity burned a little time. Now production is fast, and the same vague requirement produces a complete implementation of the wrong thing in an afternoon. The rework is not a correction to a half-built feature. It is the unwinding of a finished one.

> *An hour spent on the spec saves roughly ten hours of re-work downstream, and AI makes that ratio worse if you skip it, because the fleet produces far more implementation per hour for a reviewer to unwind.*

**10:1** The approximate ratio of rework hours saved per hour invested in the spec. AI-assisted production widens this ratio because the fleet produces more implementation per cycle for a reviewer to unwind.

Consider the math concretely. A Decision Owner spends one hour writing tomorrow's spec. That hour produces five to fifteen checkable criteria. The fleet builds against those criteria and the harness catches misalignments during the build, before the demo, before the owner has to spend time judging something that was never going to pass. Without that hour, the fleet implements against a verbal description or a chat thread. It produces something that looks right and might be right. The owner discovers at demo that three of the eight behaviors are wrong. The next day is spent unwinding and rebuilding. One day of production gained, one day of rework spent. Net: zero.

The organizations that resist the spec investment tend to be the ones that have never measured their rework rate. When they do, they discover that 30 to 40% of their production time is spent rebuilding things that were built to the wrong specification, or to no specification at all. The spec does not slow them down. The absence of the spec already has.

The strongest objection is predictable: "We move fast and iterate. A spec adds ceremony." Iteration without a spec is not iteration. It is wandering at speed. Iteration means changing direction based on evidence from the last cycle. Without a spec, there is no contract to evaluate the cycle against, and "iterate" becomes a synonym for "rebuild until someone stops objecting." Teams that say they iterate without specs are teams whose rework rate is hidden inside their velocity.

The spec is not a document for its own sake. It is a testable contract, written before the build starts, that defines what "right" means in terms precise enough for a machine to verify. Every behavior is stated as "when X, then Y" so that each clause becomes a check. Every acceptance criterion is independently verifiable. The spec is not a wish list. It is a contract that the harness enforces.

## The Volume Answer

This is the most common practical question, so it gets a precise answer.

**1 page** One day's spec is one increment, roughly five to fifteen acceptance criteria, and it fits on one page.

If it does not fit on a page, it is two specs. If it has more than about fifteen criteria, the fleet cannot complete and prove the increment in a single build day. You will end the day with a partial increment and no clean acceptance decision.

The discipline is in the constraint, not in the content. A one-page spec forces the author to choose the essentials. A five-page spec lets them defer that choice, and the deferral is what produces the oversized increments that span multiple days and resist clean acceptance.

One increment. One page. One day. If any of the three is violated, the other two follow.

The decision rule is concrete. If the spec exceeds one page, split it into two increments before starting the build. If you cannot split it, the outcome brief is not specific enough, and that is a conversation for daily planning, not a reason to write a longer spec.

## The Time Answer

**45–90 min** The Decision Owner's daily time budget for spec work. If three hours, something is wrong.

If the Decision Owner is spending three hours on a spec, one of three things has gone wrong.

- **The slice is too big.** The increment carries more behavior than a single day can build and prove. Halve it.

- **The spec agent is misused.** The owner is typing prose that a machine should draft. The agent produces the structure, restates the context, and proposes criteria. The owner corrects and signs.
- **The outcome is under-specified.** The level above handed down a brief that is not decision-ready. The owner is doing strategic work that should have been done at intake.

Three hours is a diagnostic, not a budget overrun. It means the input to the spec session is wrong, and spending more time in the session will not fix it.

If the Decision Owner is spending more than 90 minutes, apply this decision tree. First, count the acceptance criteria. More than fifteen means the slice is too big: halve it. Fifteen or fewer but the session still ran long means the spec agent is underused: check whether the owner is drafting prose the agent should produce. If the agent is being used and the criteria count is right, the problem is upstream: the outcome brief from the Flow Council is not decision-ready.

## The Authorship Split

The Decision Owner does not write a spec from a blank page. The time budget comes from a deliberate division of labor.

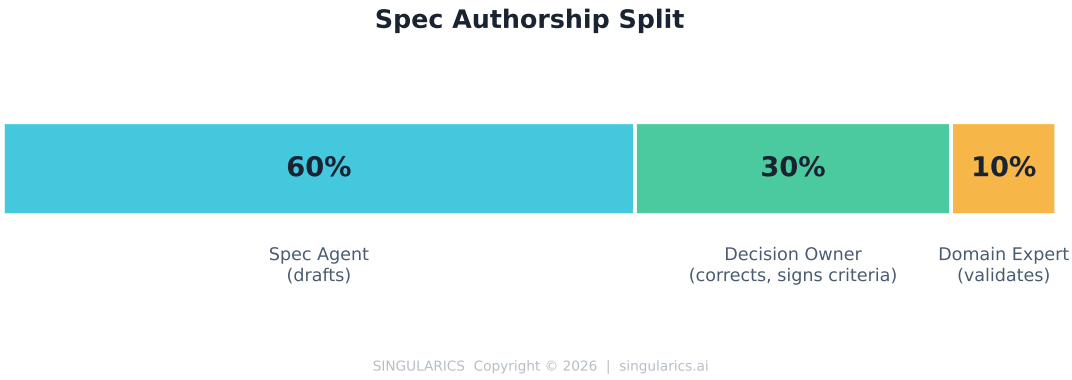


**Figure 1:** The spec authorship split. The spec agent drafts the structure and proposes criteria. The Decision Owner corrects, adds domain constraints, and signs every criterion. The domain expert validates correctness in a focused 15-minute review.

**The spec agent drafts.** It produces the structure, restates the context, proposes acceptance criteria from the outcome brief and the previous increments, and flags what it does not know. A usable draft arrives in minutes. The agent is one of three distinct agent roles in the fleet.

**The Decision Owner corrects and signs.** They fix what the agent got wrong, delete what is noise, add the constraints only a domain person would know, and personally approve every acceptance criterion. The criteria are the one part the owner must author in substance, because the criteria are the judgment. Everything else can be machine-drafted.

**The domain expert validates.** A member of the Domain Knowledge Network reviews the criteria that touch their specialty. This is a fifteen-minute review, not a co-authoring session. They are checking correctness, not style.

> *If the Decision Owner is typing more than they are deciding, the balance is wrong.*

The governing ratio is effort spent deciding versus effort spent producing text. The spec agent handles production. The Decision Owner handles judgment. When those two activities blur, the session runs long and the owner's attention drifts from the questions that matter to the formatting that does not.

In practice, the split settles within two weeks. A new Decision Owner's first spec session runs 90 minutes and the agent contributes perhaps 40% of the draft. By the fifth session, the agent contributes 70 to 80% and the session takes 45 minutes. The learning curve is steep because the skill is curation, not composition, and curation transfers faster than writing does.

## Inside a One-Day Spec

A spec for a single day's increment contains seven elements. Each one earns its place by preventing a specific class of failure.

1. **The outcome this increment serves,** in one sentence, traced to the current horizon. This is not a feature name. It is the business result the increment moves closer. Tracing to the horizon keeps the team connected to why the work is wanted and prevents increments that are technically interesting but strategically irrelevant.
2. **What exists already,** so the fleet does not rebuild it. Prior increments, existing interfaces, data already modeled. Omitting this produces duplicate work on day one and architectural drift by day five. The Decision Owner maintains this section across days; the spec agent updates it automatically from the previous increment's landing record.
3. **The behavior being added or changed,** stated as "when X, then Y" so each clause becomes a check. This is the body of the spec and carries the most

acceptance weight. Every clause maps directly to a harness check, which is why the phrasing discipline exists. "The system should handle edge cases" maps to nothing. "When the input date is before the origination date, the system returns a 422 with the message 'Date precedes origination"' maps to exactly one check.

4. **Acceptance criteria,** each independently checkable, each signed by the owner. A criterion that cannot be checked is a criterion that cannot be accepted. Rewrite it until the check is obvious. The Decision Owner signs each criterion individually, not the spec as a whole. This forces them to engage with each one rather than scanning the page and approving in bulk.
5. **Constraints and policy the increment must not violate.** These are the guardrails. They come from the domain expert and from the envelope the Intent Council set at the start of the outcome. Constraints are stated as "must not" rather than "should not," because the harness enforces them in the Control layer and a "should not" cannot be enforced.
6. **Explicitly out of scope.** This is the cheapest defect prevention available. Naming what the increment will not do saves more time than naming what it will, because scope creep in a one-day cycle is fatal to acceptance. If the fleet builds something out of scope, no matter how useful, the increment cannot be accepted cleanly because the harness will not cover it and the owner did not sign criteria for it.
7. **Open questions the owner expects to answer during the day.** These are the known unknowns. Stating them up front means the fleet can proceed on the rest while the owner resolves them, rather than discovering them at 3pm and losing the build day. A spec with zero open questions is either a simple increment or a spec whose author has not thought hard enough about what could go wrong.

Every element serves a specific audience. The outcome and constraints serve the governance layer. The behavior and criteria serve the fleet and the harness. The existing state and open questions serve the Fleet Lead who decomposes the spec into executable work. Out of scope serves everyone, because it is the element most likely to prevent a wasted day.

## The Spec Pays for Itself

The objection arrives on schedule, usually in the first week. "We are spending time on the spec that we could spend building."

The objection treats the spec as a tax on production. It is not. It is an investment that compounds within the same day.

**1 hr** of spec work saves roughly 10 hours of rework downstream. AI makes the ratio worse if you skip the spec, because the fleet produces far more implementation per hour for a reviewer to unwind.

Without a spec, the fleet implements against a verbal description or a chat thread. It produces something that looks right and might be right. The Decision Owner reviews it at demo, discovers that three of the eight behaviors are wrong, and the team spends the next day unwinding and rebuilding. One day of production gained, one day of rework spent. Net: zero.

With a spec, the fleet implements against a contract. The harness catches two of those three misalignments during the build, before the demo, before the owner has to spend time judging something that was never going to pass. The third misalignment surfaces at demo and is corrected in the next day's spec. One day of production, one clean acceptance. Net: one shipped increment.

> *Speed of production raises the cost of ambiguity. The spec pays down that cost before it compounds.*

At one-day cycles, even a modest investment in the spec pays for itself before lunch. Concretely: one hour of spec work eliminates an average of 1.5 rework days per week in the teams we have observed. Over a quarter, that is roughly 18 days of recovered production per team, which is more than a full month of additional output for an investment of 20 hours of the Decision Owner's time.

## The Day-Ahead Tension

State the tension plainly. The spec runs a day ahead so the build day starts ready. But the Decision Owner often does not know what they want next until they have seen today's demo. Those two facts appear to contradict each other.

They do not, for four reasons that compound.

### Most of tomorrow's spec is already known

Separate every spec into two parts.

The **stable part** is the outcome, the constraints, the policy envelope, the data model, the acceptance philosophy, and the general shape of the next several increments. This is written days or weeks ahead and changes rarely.

**70–80%** of the spec's substance is stable: constraints, data model, acceptance philosophy. Only 20–30% is volatile: which slice comes next and its exact criteria.

The **volatile part** is which specific slice comes next and its exact criteria. This is what the demo informs. Once you see the split this way, daily planning is not writing a spec from scratch in twenty minutes. It is making a small delta decision against a mostly-written contract.

### The buffer is a queue, not a fixed lead time

Do not think of the spec as written exactly one day early. Think of it as a shallow queue of one to three candidate specs, kept ready.

Most days the demo confirms the direction and you pull the next queued spec unchanged. Some days the demo surprises you and you reorder the queue or amend the spec at the top. The queue absorbs volatility that a rigid one-day lead time cannot.

Keep it shallow. A queue deeper than three specs is a backlog forming, and it means you are speculating further ahead than your learning rate justifies.

### Daily planning makes one of three calls

This is the operational mechanic. At daily planning, after the demo, the group makes exactly one decision about tomorrow.

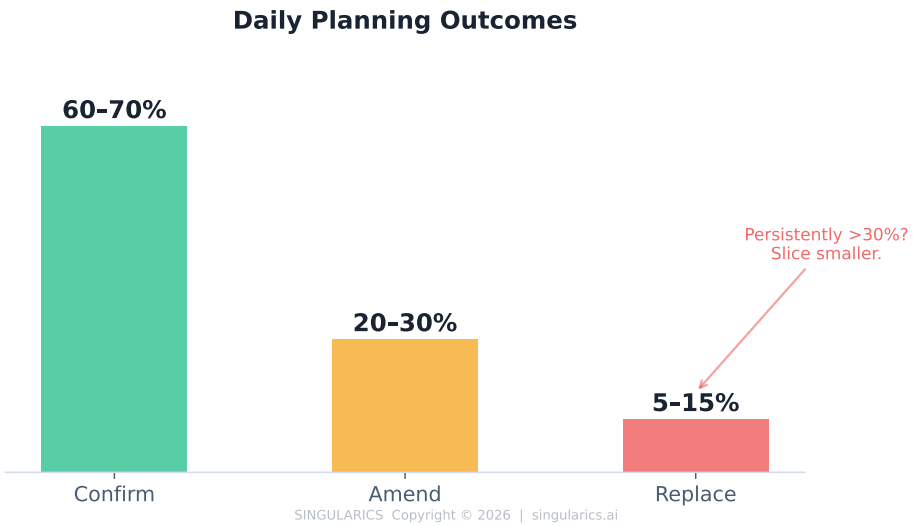


**Figure 2:** The three outcomes of daily planning. Confirm should be the majority of days. A persistently high Replace rate is a diagnostic: the slices are too big or the outcome is exploratory and the cadence should change.

**Confirm.** The demo went as expected. Pull the next queued spec as written. This should be the majority of days, roughly 60 to 70% in a healthy team. The Decision Owner does nothing to the spec. The build day starts clean.

**Amend.** The demo revealed something that changes the next increment's criteria but not its direction. The Decision Owner adjusts the top spec that evening or first thing next morning, typically 20 to 30 minutes of work. Expect this 20 to 30% of days. The stable part of the spec stays intact; only the volatile part changes.

**Replace.** The demo invalidated the plan. The queued spec is wrong and a new one is needed. Expect this 5 to 15% of days. A Replace is not a failure. It means the team learned something expensive early, which is the entire point of shipping daily.

## The design rule underneath

> *If every demo overturns the plan, your slices are too big.*

A demo that invalidates a week of planned work means you planned a week ahead of your knowledge. A demo that adjusts tomorrow means you sliced correctly.

The Replace rate is a diagnostic. Persistently above 30% means slice smaller, or admit the Effort is exploratory and change the cadence. Persistently at 0%

means you are building something so well understood that you should question whether it needed this model at all.

Three ranges tell the story:

- **Replace at 5–15%.** Healthy. The team is learning at the right grain. Surprises happen, but the queue absorbs them.
- **Replace at 15–30%.** Caution. The slices may be too large, or the outcome is entering unfamiliar territory. Check whether the stable part of the spec is truly stable.
- **Replace above 30%.** The cycle is not functioning as designed. Either halve the slice, convert to an explicit exploratory cadence, or acknowledge that the outcome needs more strategic clarity before daily production is viable.

## When You Don't Know, Spend a Cycle Finding Out

Replace days are not failures. They mean you learned something expensive early, which is the entire point of shipping daily.

When a Replace happens and the new direction is not yet clear, do not force a build day. Run tomorrow as a **spec-and-explore** cycle. The fleet investigates, prototypes throwaway options, and gathers what the owner needs to decide. The day still ends with something in someone's hands, but the artifact is a decision rather than an increment.

> *A spec-and-explore day is the "learn" branch of ship-or-learn. The artifact is a decision, not an increment. It is a legitimate, planned outcome rather than a lost day.*

The discipline is in the ending, not the beginning. A spec-and-explore cycle still closes with a named output: a decision, a revised direction, a set of constraints that were not visible before. What it does not produce is an increment, and that is fine.

An organization that treats every non-shipping day as a failure will suppress the learning that prevents the failures that actually matter. The spec-and-explore day is how the model handles genuine uncertainty without abandoning the daily discipline.

## The Failure Modes

Two failure modes break spec-driven development. Each has a tell and a correction.

### Spec as afterthought

The team starts building while the spec is still open. The Fleet Lead decomposes work against a draft because the Decision Owner has not finished the criteria. The build runs ahead of the contract.

**The tell:** the build day starts before the spec is signed. Sometimes this is explicit: "We'll start on the parts we know and finish the spec later." Sometimes it is quiet: the Fleet Lead reads yesterday's spec and assumes tomorrow's is the same.

**The correction:** a hard gate. No build day starts without a signed spec. This is not bureaucracy. It is the minimum condition for the harness to have something to check against, and for the Decision Owner to have something to accept against. A team that builds without a spec is a team whose acceptance decision is "does it look right," and that decision is how demo-driven acceptance replaces evidence-based acceptance.

When the hard gate causes friction, the friction is diagnostic. It means the spec session is running too long, the slices are too big, or the Decision Owner is not released at the committed level.

### Slices too big

The spec describes more behavior than a single day can build and prove. The increment spans two or three days. Acceptance cannot happen cleanly because the harness is red on the criteria that were not reached yet.

**The tell:** the Replace rate is persistently above 30%, or increments regularly span more than one day. The Decision Owner starts saying "we'll finish this tomorrow and accept then," which is the sound of the daily cycle becoming a multi-day cycle.

**The correction:** halve the slice. Re-run the checkability test on every criterion. If a criterion takes more than a few hours to build and prove, it is carrying too much behavior and should be split into two criteria, each of which the fleet can complete in half a day. The volume answer (one page, five to fifteen criteria, one day) is a constraint that exists precisely to prevent this failure mode. Enforce it.

**Building fast but reworking faster? We can help.**

30 minutes to talk about spec-driven development. No pitch.

Book a call | singularics.ai

## SINGULARICS

We help large organizations close the intent gap between strategy and execution so that when they accelerate with AI, they accelerate in the right direction.

singularics.ai | Book a conversation