How Many Things Can You Finish?

The limiting factor on multitasking, and why it moved.

Working draft. Simulation plus a single-subject archive study, n=48. Companion code and data: `delusion/sim`, `delusion/data`.

Abstract

An attempt requires two distinct efforts: `e_build`, to make a working artifact, and `e_ship`, to put it in front of anyone. Both draw on one finite budget. I show that the optimal number of simultaneous attempts depends on their *sum*, while the intuitive heuristic most builders use depends on `e_build` alone. Historically the two agreed closely, because building dominated. Agent-assisted development collapsed `e_build` by roughly an order of magnitude and left `e_ship` untouched, and the two prescriptions have diverged.

In simulation, as building gets cheaper the optimal attempt count rises, from 4 to 48 across the range studied, while shipped output saturates, and the success probability under build-capacity planning falls by more than two orders of magnitude, from 0.163 to 0.0006. The penalty for the old heuristic grows at exactly the moment the heuristic becomes most tempting. A closed-form prediction for the size of that error is derived and then falsified by simulation; cost heterogeneity softens it substantially.

I test the mechanism against a single-subject archive of 48 repositories over 21 months with contemporaneous status notes recorded before outcomes were known. Of 17 projects the notes mark as stalled, 100% are stalled at a step that building cannot resolve, and 0% are stalled purely on more code. A strong concurrency effect is found and then declined as unidentifiable. I propose that the relevant sense of self-delusion here is not a probability error but a *category error about which threshold effort is paying down*.

1. Introduction

Two pieces of advice are routinely given to people who build things, and they contradict each other. Be calibrated: most attempts fail, so do not overrate yours. And be relentless: the people who succeed are the ones who kept starting things. Both are supported by evidence, which usually gets resolved by declaring it a matter of temperament.

It is not a matter of temperament. It is one inequality read on two sides of a threshold, and the threshold has moved.

The starting observation is that an attempt is not one activity. Getting something to work and getting it in front of a person are separate efforts with separate costs, and they are not interchangeable. Call them `e_build` and `e_ship`. For most of the history of software `e_build` was much the larger of the two, which had a convenient consequence: you could plan your workload by asking how much you could build, and be approximately right about how much you could ship.

Agent-assisted development broke that. It reduced `e_build` by something like an order of magnitude while leaving `e_ship` essentially fixed, because shipping is largely composed of things that do not compress: a credential another party issues, an app review on someone else's clock, a registration with a human in the loop, a decision only you can make, a customer who answers when they answer. This paper works out what that does to the optimal number of attempts, and to the accuracy of the felt sense of progress.

The contribution is threefold. First, a two-threshold attempt model in which the optimum depends on the sum of the thresholds while the common heuristic depends on one of them, so that the error between them scales inversely with build cost. Second, a simulation that quantifies the divergence and, in the process, falsifies the closed form I derive for it. Third, a single-subject empirical study whose distinctive feature is that the blockers were written down before the outcomes were known, which is unusual in this literature and removes the largest source of reconstruction bias.

2. Related work

Four literatures border this one, and it is worth being precise about what is already occupied.

Search-adjusted inference. When many hypotheses are tried and the best is reported, its apparent quality is inflated. The Deflated Sharpe Ratio is the canonical treatment in finance, and the garden of forking paths is the general form. My own prior work extends this to hypotheses proposed by language models, where the effective trial count is not directly observable. That work asks how much to *discount* a result given the search. This paper asks a different question: how much search to *do*.

Motivated belief. Overconfidence has been modelled as a self-commitment device with real instrumental value, which treats the bias as a bias that happens to pay. The framing here is different: with belief-dependent effort, the rational credence is a fixed point rather than a point

estimate, and that fixed-point equation can admit multiple stable solutions. Optimist and pessimist can each be accurate about the world their own belief produces, which makes the disagreement empirically irresolvable from inside either one.

Non-ergodicity. The argument that time averages and ensemble averages diverge under multiplicative dynamics supplies the boundary condition this paper needs but does not extend: a high-belief equilibrium is only reachable if the transit is survivable, so ruin probability, not expected value, bounds rational persistence.

Entrepreneurial experimentation. The option-value view of attempts, in which each buys information rather than an outcome, is well developed. What is missing there, and is the gap this paper occupies, is that the cost of purchasing an option has fallen sharply and *asymmetrically*, and that the asymmetry alone changes the optimal policy even with all beliefs held fixed.

3. The model

3.1 Setup

An agent has a finite budget `E` of non-delegable attention per period and starts `M` attempts. Each attempt carries two costs drawn from independent lognormal distributions: `e_build`, the effort to reach a working artifact, and `e_ship`, the effort to reach a user. Both consume `E`, and an attempt ships only if both are paid.

Shipping also involves waiting on third parties. That is modelled as latency rather than effort, because it consumes no `E`. It matters for calendar time and for mortality, not for allocation, and it is deliberately excluded so that the allocation result is not doing its work through a queueing artifact.

Write `r = e_build / e_ship` for the ratio of mean costs. Agent assistance is a reduction in `r`. The pre-agent regime is `r > 1`; the current regime is `r < 1`.

Attempts are built in the order they are started until the budget is exhausted, then whatever remains is spent shipping, cheapest first. Cheapest-first is a deliberately generous assumption: it gives the widening strategy the best possible chance, so any penalty it suffers is a lower bound. Each shipped attempt succeeds independently with probability `p`, and the objective is the probability of at least one success, `1 − (1 − p)^N_shipped`, maximised over `M`.

3.2 Two policies

The optimal policy allocates against the total cost of a completed attempt. The intuitive policy, the one that answers the question "how many of these can I build?", allocates against build cost alone.

```
optimal      M*       proportional to  E / (e_build + e_ship)
heuristic    M_build  =                E / e_build

homogeneous-case ratio:  M_build / M*  =  1 + e_ship/e_build  =  1 + 1/r
```

This is the central structural claim, and it explains why the heuristic was ever any good. At `r = 3`, where building cost three times what shipping cost, the two prescriptions differ by 33% and the heuristic is a serviceable approximation. At `r = 0.1` they differ by a factor of eleven. The heuristic did not become wrong because people got worse at reasoning. It became wrong because the quantity it neglects went from negligible to dominant.

3.3 What the closed form gets wrong

The expression above assumes homogeneous costs. It does not survive contact with the simulation, and the direction of its failure is informative. The closed form predicts the error ratio grows from 1.33 to 11.00 across the range studied, while the simulated ratio grows only from 1.75 to 4.17.

The reason is that with heterogeneous costs and cheapest-first shipping, a widened portfolio is not uniformly penalised. Starting more attempts also samples more of the low-cost tail of the shipping distribution, and some of those get shipped almost for free. Heterogeneity is therefore a partial hedge against over-widening, and the closed form should be read as an *upper bound on the error* rather than an estimate of it. I report it because being wrong about the magnitude while right about the mechanism is the useful kind of wrong, and because the bound is what a reader would derive first.

4. Simulation results

Parameters: `E = 20` in units of mean shipping cost, sigma 0.45 lognormal on both costs, `p = 0.08` per shipped attempt, 20,000 Monte Carlo draws per cell, `M` searched over 1 to 90.

r	M*	shipped at M*	P at M*	M_build	shipped at M_build	P at M_build	error ratio	closed form
3.0	4	3.6	0.259	7	2.23	0.1630	1.75	1.33
2.0	6	5.1	0.345	10	1.91	0.1406	1.67	1.50
1.0	10	8.4	0.500	20	1.11	0.0844	2.00	2.00
0.5	15	12.0	0.630	40	0.50	0.0391	2.67	3.00
0.3	21	14.7	0.704	67	0.21	0.0163	3.19	4.33
0.1	48	20.7	0.821	200	0.01	0.0006	4.17	11.00

Table 1. As building gets cheaper the optimal attempt count rises steeply and success under optimal allocation improves from 0.26 to 0.82. Under build-capacity planning it collapses from 0.163 to 0.0006. The two policies coincide near parity and diverge in both directions from it.

Three results, in order of how much weight they can bear.

Result 1. The optimal attempt count genuinely rises. From 4 to 48 as `r` falls from 3.0 to 0.1. This deserves emphasis because it is the opposite of a scolding: cheap building really does mean you should start more things, and "focus on one thing" is bad advice in this regime. The relentless camp is right about the direction.

Result 2. Shipped output saturates. From 3.6 to 20.7, sublinear throughout, asymptoting on `E / e_ship`. The ceiling on how much reaches a person is set entirely by the threshold that did not fall. Nothing done to build capacity raises it.

Result 3. The penalty for the old heuristic grows as the heuristic becomes more attractive. Success probability under build-capacity planning falls from 0.163 to 0.0006, a factor of 270. This is the result with teeth. The regime that makes it feel possible to start forty things is the same regime in which starting forty things is worst.

4.1 Robustness and a real boundary

Sweeping `E` over {10, 20, 40}, `p` over {0.03, 0.08, 0.20} and sigma over {0.25, 0.45, 0.80}, both claims hold jointly in **21 of 27 configurations**. All six failures share a signature: sigma 0.80 with `E` at most 20, where the optimum is censored at the search ceiling.

> **Boundary condition, not noise.** At high cost heterogeneity and small budget, the optimal policy stops being "choose the right number of attempts" and becomes cost arbitrage: start very many, ship only the cheapest tail, and ignore everything else. In that regime widening is correct almost without limit and Result 1 has no interior optimum to find. This is a genuine regime rather than a modelling artifact, and it identifies who the paper does not apply to. If your attempt costs vary by more than roughly a factor of ten and your budget is small relative to them, spray. The prescription here holds for moderate heterogeneity, which is the common case.

5. Empirical study

5.1 Data

A single subject's complete project archive: 51 repositories, of which one has no readable history and two are clones of others, leaving **n = 48** spanning November 2024 to August 2026. For each, full commit history yields first and last activity, commit count, distinct active days, and lines changed. Alongside it sits a set of 74 contemporaneous status notes, written during the work and before outcomes were known.

That last property is the reason this archive is worth studying despite n=48 and a single subject. Work on motivated belief is usually forced to reconstruct prior beliefs after outcomes are known, which contaminates precisely the variable of interest. Here the blockers were written down at the time.

> **A measurement problem that must be reported first.** Commit-derived effort measures are *not valid* for this subject's recent work. Inspection of first commits shows entire subsystems arriving at once: one project's initial commit is 137 files and 12,916 insertions; another's complete history is a single commit of 76 files made on the day the work finished. Where a project is largely agent-built in one session, the repository is often initialised at or after completion. First-commit date is therefore not a start date, and commit counts understate effort severely. Any study of this kind that uses commit volume as an effort proxy without checking this will conclude that recent projects were barely worked on. Several were built intensively in hours.

5.2 Primary result: what stalled projects are stalled on

Coding the notes by blocker type gives the study's main finding. Initial automated coding returned 18 stalled projects; audit against the full notes found one false positive, a regular expression matching `PARKED` inside the string `UN-PARKED`, a negation read as a confirmation. Corrected, n = 17. Three ambiguous cases were re-coded by hand with written reasons.

Blocker category	n	share	Examples from the notes
Third-party external	14	82%	An API key, a testnet faucet, agent registration, marketplace approval, an APNs auth key, server provisioning, a domain
External and build	2	12%	A missing frontend plus an undeployed backend; a conversion pending plus a go-ahead
Own non-delegable step	1	6%	A one-week validation test only the subject can run. No third party, no code
Purely more code	**0**	**0%**	None
Involves a non-automatable step	17	100%	Every stalled project without exception

Table 2. Blocker classification of 17 stalled projects from contemporaneous notes. The finding strengthened after an audit that corrected an error *against* the hypothesis.

Only 13 of 48 projects, 27%, ever received sustained iteration past their initial build. Every one of the 17 the notes mark as stalled is stalled at a step no amount of building resolves. Not one is waiting on code.

5.3 A strong result I decline to claim

Projects begun while many others were active were far less likely to receive sustained attention. Restricting to a 90-day minimum observation window to handle right-censoring, Spearman between concurrency and post-burst active days is **−0.60**, with 82% sustained in the low-concurrency half against 25% in the high.

I do not claim it. Splitting by era shows why.

Era	n	concurrency range	sustained	mean days after burst
before 2026-03	8	1 to 8	7 / 8	30.8
2026-03 to 2026-04	12	9 to 25	6 / 12	7.8
2026-05	7	23 to 26	0 / 7	0.4

Table 3. Concurrency does not vary meaningfully within eras, because the subject accumulated projects monotonically. Concurrency and calendar time are the same variable in this dataset and cannot be separated.

There is no low-concurrency 2026-05 project to compare against. Spearman between concurrency and start date is +0.35, and any apparent effect is partly a time trend. Worse, a specific rival explanation is credible given the measurement problem above: post-burst inactivity may indicate a project finished in one session rather than one starved of attention. A correlation of −0.60 that cannot be decomposed is not evidence, and reporting it as such would be exactly the error this paper is about.

5.4 Calibration, and what is not identified

Two observations cannot identify three parameters. Normalising `E` in units of shipping cost and fixing `M = 48`, the value of `r` reproducing 13 shipped is 0.111 at `E` = 14, 0.148 at 16, 0.223 at 20, and 0.336 at 26. That is, shipping costs between three and nine times what building costs for this subject. The point estimate is not identified; the order of magnitude is. The observed `M` = 48 sits between `M*` and `M_build`, not at either extreme, so the subject is partially but not fully captured by the heuristic.

6. Discussion

6.1 Delusion as a category error

The framing I began with treated self-delusion as a probability error: credence exceeding what the evidence licenses. The data do not support that as the operative failure here, and something more specific fits better.

Building *feels* like progress toward shipping. For most of the history of the craft that feeling was a decent estimator, because the two thresholds moved together and building was the larger of the two. When `e_build` collapsed and `e_ship` did not, the felt sense of progress decoupled from actual progress. A belief calibrated to the old ratio became systematically wrong without ever feeling wrong, because the sensation it was tracking is still being generated at a higher rate than before.

> The delusion is not about probability. It is a category error about which threshold effort is paying down.

This is testable in a way a probability-miscalibration story is not. It predicts that stalled work clusters at the non-automatable step, which Table 2 finds at 100%, and that the felt-progress signal correlates with build activity rather than with distance to a user.

6.2 Predictions

- **Aggregate inventory.** Public repository counts per developer should have risen sharply since agent assistance became common, while shipped-artifact counts should be roughly flat. Result 2 says the ceiling did not move.
- **Blocker composition shift.** The share of abandoned projects blocked on non-code steps should rise over time within individual histories, not merely across them.
- **Domain gradient.** The effect should be strongest where `e_ship` is most rigid. Regulated software, app stores, hardware and enterprise sales should show the largest gap; a personal script published to a package registry the smallest.
- **Intervention.** Capping concurrent attempts at the shipping-constrained optimum rather than the build-constrained one should raise shipped output without raising effort. This is directly testable prospectively and is the obvious next study.

6.3 Limitations

Stated plainly, because several are severe. **n = 48, one subject, no control.** Nothing here establishes generality; the archive supports a mechanism, not a population estimate. **Blocker coding is partly manual** and performed by a party to the analysis; the audited reasons are published so the coding can be contested, but it is not blind. **The concurrency channel is unidentified** and is reported only to record that it was looked for. **Success probability `p` is assumed independent across attempts**, which is wrong in the direction of understating the value of concentration, since real attempts share learning. **Effort is treated as a single fungible budget**, when in practice building and shipping draw on partly different capacities. **The closed form in section 3.2 is falsified** by my own simulation and retained only as a bound.

The strongest claim the evidence supports is the composition result in Table 2. Everything about magnitudes is model output, and the model is a first draft.

7. Conclusion

Be calibrated and be relentless are not opposing temperaments. They are the same inequality evaluated on two sides of `E / (e_build + e_ship)`, and the reason the advice feels

contradictory is that most people are computing it with one term missing.

Cheap building genuinely means you should start more things. That part of the optimistic instinct is correct and the simulation supports it strongly: the optimal attempt count rises more than tenfold across the range studied. What it does not mean is that you should start as many as you can build. That number is now three to eleven times too large, and the gap widens every time building gets cheaper.

The uncomfortable version, and the one the archive actually supports: forty-eight repositories is not evidence of a lack of discipline. It is close to a rational response to a real collapse in the cost of an attempt. The error is subtler and more correctable than a character flaw. It is having re-optimised against the threshold that moved, while the one that binds stayed exactly where it was.

Appendix. Reproduction

All simulation figures come from `sim/fast.py` at a fixed seed. A slower, independently written reference implementation lives in `sim/model.py`; agreement between the two is the only check performed on the simulation code, and a third implementation would be worth having.

```
data/projects_raw.json       51 repos, raw git statistics
data/projects.json           48 after dedupe, concurrency + sustained
data/labels.json             initial automated blocker coding
data/labels_audited.json     corrected, audit_reason on every change
data/stalled_audit.json      the 17, with full note text
AUDIT.md                     1,902 lines of evidence behind each call
scripts/extract.py           git history extraction
scripts/analyze.py           concurrency and sustained-iteration measures
scripts/fixed.py             censoring and confound checks
scripts/label.py             blocker coding rules
sim/model.py                 reference implementation, readable
sim/fast.py                  vectorised, used for all reported figures
sim/calibrate.py             Table 1, Table 4, robustness sweep
```