

FIELD NOTES FOR THE COO · DEPLOYMENT STRATEGY

When Exceptions Outrun the Desk

Why exceptions the desk cannot clear leave the procedure, and four numbers to watch each week before the record drifts

---

Marcos Stipicich
Deployment Strategist · 13 years in supply chain and operations

Cover: Adolph Menzel, The Iron Rolling Mill, 1872-1875

The argument in one page

When exceptions reach the desk faster than its people can resolve them, the overflow does not stop the dock. It leaves the procedure.

It surfaces weeks later as dirty data, and it is often answered with a reminder about discipline. The arithmetic suggests otherwise. Exceptions arrive at the rate the trucks bring them, and the desk clears them at the rate its people can. When the first rate is higher, the queue grows every shift and something has to give. The dock has a truck waiting; the inventory record has until month-end. A business case that takes a person off the desk without taking exceptions off the dock has already chosen which one gives.

1. **The pressure fits in one ratio:** the exceptions that arrive in a shift over the ones the people on the desk can clear. Above 1, the queue grows every shift.
2. **The desk turns slow before it is full.** Trucks bunch and exceptions vary, so the wait climbs steeply while the averages still look fine.
3. **So keep four numbers a week,** and read them before the next headcount decision and before the new system writes more to the record on its own.

The arithmetic of the desk

Every shift brings exceptions the system cannot resolve on its own: a pallet with no label, a count that does not match the advance notice, a line that is not on the order.

Let λ be the receipts that reach the dock in a shift (a receipt is one delivery line), p_e the share that need a person, h how many exceptions one person resolves in a shift by the procedure, and n the people actually available for them on that shift. The pressure on the desk, which my manuscript calls bypass pressure, is:

$$B = \lambda \cdot p_e / (h \cdot n)$$

The top is what arrives in a shift; the bottom is what the desk can clear. B is how full the desk runs (queueing theory calls this utilization). Above 1, the queue grows every shift by the difference, and the procedure alone will never clear it. At exactly 1, if arrivals vary, it does not settle either.

Below 1 is no guarantee. Trucks arrive in bunches, and one exception takes two minutes while the next takes an hour. In the heavy-traffic approximation John Kingman published in 1961, the wait grows with the load divided by the spare capacity, and with how uneven arrivals and service are. With three people and random arrivals and service, the average wait in the queue is about one service time at 0.8, nearly three at 0.9 and six at 0.95.

Forty a week

Take an invented receiving desk, with round numbers. A shift brings 300 receipts, and 8 in every 100 need a person. One person clears 10 a shift by the procedure, and three people work the desk. Twenty-four exceptions arrive; thirty can be cleared. B is 0.8: busy, and already worth watching.

The business case for the new system takes one person off the desk, because the system will resolve most exceptions on its own, according to the slide. For two people to keep up even on average, the system would have to cut that share from 8 in 100 to below about 6.7. On this dock, it does not: twenty-four still arrive, and the desk can now clear twenty.

B is 1.2: four more than the desk can clear, every shift. Over a week of ten shifts, forty, and the pile does not shrink on its own.

Something has to give. The dock can stop. The desk can clear the pile on overtime, and the head the business case saved comes back on the overtime line. Or part of the queue leaves the procedure: pallets received against the advance notice, counts keyed to match, differences sorted out later. A truck waits only so long, and overtime has a budget. Past both, the record is what gives.

Nobody at that desk chose to dirty the record. The business case did the sums and left out where the exceptions go.

Four numbers a week

NUMBER	HOW TO GET IT, AND WHAT IT TELLS YOU
λ: receipts in a shift	At the gate, not in the system: the system counts what it received, the gate what arrived. When the desk falls behind, the difference waits in the yard
p_e: share that needs a person	From the exception log, per thousand receipts, beside the twelve-month rate. If it drops and nothing changed on the dock, ask where the exceptions went
h: cleared per person in a shift	Time twenty real exceptions, by the procedure, with whoever resolves them. Divide the productive minutes in a shift by the average minutes per exception. A shortcut's speed is borrowed from the record
n: people on the desk	Who is actually there on the shift, not the roster. Whoever is unloading a late truck is not on the desk

Work out $B = \lambda \cdot p_e / (h \cdot n)$ for your busiest shift, not the weekly average: the average can read 0.8 while Monday reads 1.1. One check needs no formula: count the open exceptions at the end of every shift, and the desk's overtime hours. If either grows week after week, the week as a whole is above 1, whatever the sheet says.

Use it as a go-live gate: do not extend what the new system writes on its own until B is back under 1 on the busiest shift. Above 1 for two weeks in a row, the workaround is a matter of time. Between about 0.8 and 1, the desk keeps up on paper and falls behind at the peaks: watch how long exceptions wait.

What moves B. Fewer exceptions at the source, a faster correct path at the desk, or more people on it. Overtime is more people at a higher rate, on a line the business case said it had saved.

What this does not prove

- **B is a model, and 1 is its only hard line.** Above 1 the queue grows by arithmetic. The 0.8 is a rule of thumb from my manuscript, not a result of the model.
- **Kingman worked out his approximation for a single server in heavy traffic.** The figures for three people come from the standard several-server formula (Erlang C), with random arrivals and service; trucks that bunch make them worse.
- **The desk on page 4 is invented.** Its numbers are round on purpose and come from no site.
- **That the overflow goes to the record is my reading.** It would fail on a dock where, with B above 1 for weeks and no overtime, the trucks waited in the yard and the record stayed clean.

One question before the next business case. If the new system goes live with the same exceptions and one person fewer at the desk, where do the extra ones go?

About the author

Marcos Stipicich
Deployment Strategist · Supply Chain & Operations

I grew up on a sheep ranch in Patagonia, where my father's rule was simple: reading is worth nothing if you can't see it on a working day. I still work by it: I measure before I cut, and no tool or method beats what works on the floor.

Thirteen years in Chile, Argentina and Australia taught me that supply chain is not just logistics: it is the whole pipeline, which is why it ends up putting out fires that start elsewhere. A promotion planned in marketing becomes expedited orders in purchasing and a blocked aisle in the warehouse. I would rather find where the leak starts. On one gemba walk, mixed, short and damaged orders led back to a per-operator productivity KPI tied to a bonus; I retired the metric and measured customer satisfaction instead.

I have worked most of that pipeline: planning for more than 200 stores, a three-site logistics operation, a USD 30M cross-border e-commerce P&L and a 50,000-SKU warehouse-system cutover, where I stayed on the floor until the shortcuts died and order processing time fell 20%. Deployments live or die inside that pipeline, and that is where I work. I learned to code in C; today I build with AI coding agents, keeping the math and the rollback in my hands.

Available for pre-cutover forensic audits, deployment strategy, and Forward Deployed roles.

From my manuscript, *The Deployment Chasm*.

linkedin.com/in/marcos-stipicich · marcos.stipicic@gmail.com

Sources

- Kingman, J. F. C. 1961. “The single server queue in heavy traffic.” *Proceedings of the Cambridge Philosophical Society* 57(4): 902–904; the journal is now *Mathematical Proceedings of the Cambridge Philosophical Society*. doi.org/10.1017/S0305004100036094

Image. Public domain: Adolph Menzel, *The Iron Rolling Mill*, 1872–1875. Alte Nationalgalerie, Berlin. My reading: the iron sets the clock. The crew works it while it glows, and the meal at the right is eaten beside the machine. On a dock, the truck sets it.

Illustration. The receiving desk on page 4 is invented to show the arithmetic; none of its figures comes from a site I worked in.