THE DELIVERY DIAGNOSTIC — EXAMPLE REPORT

Delivery & resilience audit

Six pillars assessed against your board, your retros, and your own account of how delivery works today. Two are the real bottleneck.

PREPARED FOR	DELIVERED	SCOPE
Northgate Software	8 September 2026	Intake, board, 3 retros

[01] Summary

Delivery is not failing for want of testing. Releases bundle two to three weeks of change into one irreversible deploy, and nobody is accountable for what the Definition of Done is meant to catch. Fix those two and the symptoms attributed to QA largely resolve on their own.

11
DAYS BETWEEN RELEASES

3
HOTFIXES, LAST 30 DAYS

2 of 6
PILLARS FLAGGED

HEADLINE FINDING

“Ship” and “release to users” are the same irreversible moment. Correct that process, rehearse the rollback, and you’re on your way to smoother deliveries and reduced risk.

[02] Findings, pillar by pillar

Every pillar gets the same depth of assessment. A finding panel appears only where there is something to act on. Where there is no panel, nothing was found that warrants fixing right now.

01	02	03	04	05	06
QUALITY & OWNERSHIP	TEST STRATEGY	RELEASE PROCESS	FEEDBACK LOOPS	TEAM STRUCTURE	TECH DEBT

01 Definition of Quality & Ownership

FLAGGED

In one of your retros, three people on the team gave three different answers to what “done” means. A Definition of Done exists in the wiki and is checked against by nobody. Quality is technically everyone’s job, but ownership hasn’t been defined clearly enough for each area of expertise, which in practice means it has become no one’s responsibility.

FINDING

The checklist gets ticked, but no one is accountable for what it is supposed to catch. A checklist for the sake of one is not good quality practice.

02 Test Strategy & Risk Coverage

NO ACTION

Automation covers the stable, low-change areas of the product well. The billing and entitlement flows, the highest-change / highest-cost areas, are still manually tested before each release to ensure quality coverage is in the right place.

03 Release & Deployment Process

FLAGGED

Releases currently bundle two-to-three weeks of change into a single deploy. When something breaks, there are a dozen plausible causes instead of one, which is the main reason your last two hotfixes took over a day each to isolate. The rollback runbook is written but has never been rehearsed.

FINDING

No feature flag layer exists between “code merged” and “visible to users”. Every release is an irreversible, all-or-nothing event.

04 Feedback Loops & Speed to Detection

NO ACTION

Bugs are typically caught within a day via existing monitoring, and the team maintains a reasonably low escape rate relative to its size. Detection-to-fix cycles are short, and there is no evidence of customers regularly finding issues before internal monitoring does.

05 Team Structure & Communication

NO ACTION

Team boundaries broadly match system boundaries; cross-team changes are rare enough not to be a tax. Note: Retro action items are carried without owners, so the same two items recur. It's a small friction point rather than a structural problem.

06 Technical Debt & Architecture Health

NO ACTION

Debt is felt but not written down: everyone carries an informal mental list, so it never competes fairly against feature work in prioritisation. Nothing observed in the board history suggests debt is currently blocking releases.

Two pillars carry the problem: ownership (Pillar 1) and release process (Pillar 3). The remaining four are healthy enough that effort spent on them this quarter would be effort taken from the two that matter.

[03] Prioritised fix plan

30
DAYS

01 Cut release size: reframe your planning around shipping weekly, whatever is ready.

02 Rehearse the rollback runbook once, end to end, and correct it.

03 Rewrite the Definition of Done to one page and name an owner per feature.

60
DAYS

01 Introduce a feature flag layer so deploy and release stop being one event.

02 Re-point automation at the billing and entitlement flows.

03 Start a visible technical debt register, reviewed with the feature backlog.

90
DAYS

01 Track escape rate, however roughly, and review it at retro.

02 Give every retro action item an owner and a follow-up date.

03 Reassess Pillars 1 and 3 against this report.

[04] Appendix: AI opportunities

PILLAR 01 — ACCEPTANCE CRITERIA

LLM-assisted acceptance-criteria generation at planning time closes a surprising amount of the ownership gap before code is written.

PILLAR 03 — RELEASE NOTES

Automated release-note and changelog generation from commit history removes a recurring drag on release cadence, and helps you spot patterns over time.