

Assessment Report

AI Native Assessment Report

Placing the Northvale Payments platform organisation on the re:cinq AI Adoption Ladder, and the first twelve weeks of climbing it.



prepared for

**Northvale Payments, Platform
Organisation**

prepared by

re:cinq · {think ai native}

date

**Sample edition ·
September 2026**

Document control

Client	Northvale Payments, platform organisation
Engagement	AI Native Assessment. Two days of on-site discovery, three structured workshops, a written questionnaire, fourteen recorded interview sessions, a technical review of the platform, pipelines and current AI tooling, and a delivery-data extract.
Scope	How Northvale builds and ships software. AI inside the payments products (merchant onboarding, disputes, reconciliation assistance) was raised in interviews and is a parallel workstream. It is not evaluated here beyond Section 17.
Framework	re:cinq AI Adoption Ladder, Stages 0 to 8
Version	Sample edition
Date	September 2026
Authors	Pini Reznik, Michael Müller
Classification	Confidential in a real engagement. All questionnaire and interview findings anonymised per agreement.
Evidence labelling	Every quantitative claim in this report is labelled as one of: delivery extract (measured), interview estimate, workshop estimate, questionnaire result, technical observation, or external benchmark.
Preparation	Drafted with assistance from Claude (Anthropic) for structure, wording and figures. The analysis, findings, judgements and recommendations are re:cinq's own, grounded in the discovery workshops, interviews, questionnaire and delivery data, and reviewed by the named authors.

About this sample. Northvale Payments is a fictional company. This report is a composite drawn from real re:cinq assessments of European software organisations, with every organisation, product name, figure and quotation altered so that no client is identifiable. Any resemblance to a real company is coincidental. What has not been altered is the shape of the work: the method, the evidence standard, the framework, the chapter structure and the kind of decisions a real report puts in front of leadership are exactly what an engagement delivers.

How to read this report

Each claim states its source in the sentence itself, there are no footnotes. Where a figure comes from the delivery extract, an interview estimate, a workshop estimate or our own inference, the text says which. Appendix D defines the recurring terms.

Recommendations are annotated with the transformation pattern they apply, like this (Executive Commitment, TF-11). The patterns come from the re:cinq transformation pattern library, published at re-cinq.com/pattern-library, and each one carries its own context, problem and known failure modes. Codes beginning TF are transformation patterns, AN are AI native patterns, and AP are anti-patterns the report is written to avoid. Appendix B reproduces every pattern used, with the section that applies it.

The constraint is verification, and the release window exists to hide it

{the constraint}

Northvale's main constraint is not code generation. It is that correctness is confirmed by hand, so every change to the ledger core waits for a four-weekly release window, and the certification evidence a scheme audit needs is assembled by people rather than produced by the pipeline.

Today a typical partner change takes four to five months from intake to first production. Most of that is elapsed waiting: on the partner's specification, on the compliance and risk review, and on the next release window. The hands-on work inside those waits is measured in days. The services estate already deploys on demand and the platform is in good shape. The delivery model is the problem, and the release records show its cost rising: corrective deployments have grown from a third of all deployment requests in 2024 to half in 2026 while planned deployments stayed flat.

AI is already used by everyone and one team runs an agentic set-up well ahead of the industry norm. But adoption is local and disconnected. Individuals accelerate their own work while the total delivery system does not become materially faster, and faster generation against a manual certification queue just fills the queue faster.

Around the verification constraint sits a structural one. Connectors are rebuilt per partner because the unit of reuse is too coarse, with interview estimates of 40% to 70% of connector code adapted or rewritten each time. Both are addressable. Neither gets addressed, because delivery is funded per partner programme and no single programme will pay for cross-cutting work. The transformation has to change both, and it has to be a programme with an owner, not another backlog item.

The decision required

One accountable owner for the whole journey

Analysts own part, engineering another, compliance another, QA another, the release manager another. Everyone can discharge their own responsibility while the end-to-end outcome stays slow. Nobody today is accountable for total lead time from intake to first production. The stream mirrors those functional boundaries, so changing the boundaries is the fix (Establish Team Topologies, TF-31).

Capacity taken from somewhere real

The transformation will fail if it is added to existing workloads without reducing partner commitments. Some planned delivery has to be deferred, and the platform engineering team of four has to be reinforced before it owns the factory. That is a leadership decision, and it has to be made before more tooling gets built.

One team, one change, one complete loop

One connector team takes one real, bounded partner change from approved specification to production outside the release window, and the factory gets built around it as it goes. We recommend a connector team working on a modern gateway component. It has strong existing AI usage, a certification scope small enough to automate in twelve weeks, and a verification bottleneck that is easy to observe.

The goal, stated so it can be tested

A connector team takes a bounded partner change from approved specification to production, with certification evidence produced by the pipeline and a small number of human decisions, at the same quality as today or higher.

The quality condition comes from the workshop itself, and it is not negotiable: speed in exchange for quality does not count. In a regulated payments platform that has a precise meaning. Incidents and audit findings are failure metrics. If either rises, the programme has failed whatever the lead time says.

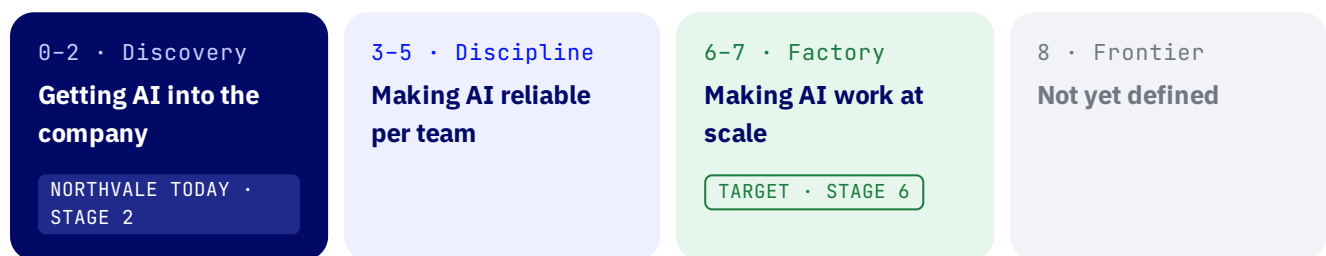
Speed needs three separate clocks, because one number invites the wrong argument. Days from approved specification to production, for a defined class of low-risk change on the services estate. Two and a half to three months from intake to first production for a typical partner change, against four to five today, which is the room's own near-term answer and depends on the partner-facing half of the journey improving with the engineering half. And a share, not a count: the low-risk change class shipping outside the release window with pipeline-generated evidence, which is the room's own test for whether the factory exists. Core and scheme-impacting changes keep their controls throughout.

What success looks like

These goals come from the goals workshop and the interviews, and they carry into the measurement section unchanged.

Goal	Today	Target
Intake to first production, typical partner change	Four to five months (interview estimate), 63 to 84 days median for the common initiative sizes (delivery extract)	Two and a half to three months near term, one month as a conditional aspiration
Release of low-risk changes	Everything touching the core waits for the four-weekly window and a one-week freeze	On demand behind flags and canaries, with the window kept only for coupled core flows
Certification evidence	Assembled by hand from test runs and screenshots, roughly three weeks per change	Produced by the pipeline as a by-product of validation, accepted by compliance

Goal	Today	Target
Corrective share of deployment requests	51% in 2026, up from 33% in 2024 (release records)	Falling, with the P1 recovery-time floor of about 70 minutes held
Connector reuse	40% to 70% of connector code adapted per partner (interview estimate)	Measured percent customisation per partner, falling as rails packs are promoted
AI adoption	24 of 24 use AI, four islands, no shared context, no measurement	One factory, one context, every team on it, adoption and outcomes measured per team
People	Audit exposure and role anxiety named as reservations	Empowered, with future roles named and the factory stated as not a headcount programme



The re:cinq AI Adoption Ladder, in one line. Nine stages (0–8) in four phases. Northvale is at Stage 2, wide individual adoption. The target is Stage 6, an AI-enabled factory with humans at the gates. The full model is explained in Section 8.

Engineering, not advice.

re:cinq is a consultancy that helps software organisations become AI native. We do not sell tools and we do not run training workshops that fade after a month. We change how software gets delivered, from intake to production, so that AI agents become a reliable part of the delivery system rather than a personal productivity trick.

Our work is grounded in delivery practice. We have built and operated production systems ourselves, and we treat AI adoption as an engineering and organisational problem, not a technology purchase. Our method starts with an assessment of how work actually flows through an organisation today, because agents amplify whatever process they are dropped into. A clean process gets faster. A broken one produces broken output faster.

A successful organisation at the limit of its delivery model

Northvale's platform organisation is doing well. It processes payments for merchants, marketplaces and partner payment providers across six European jurisdictions, its partner base is expanding, and its teams are motivated and technically strong. The gateway services estate is modern, independently deployable and already ships to production many times a day. Settlement, reconciliation and reporting run on largely shared code. AI adoption is wide and real, every questionnaire respondent uses AI tools, and one team runs an agentic set-up well ahead of the industry norm.

AI made writing code cheap. So the hard part is no longer writing. It is knowing what to build, proving it works to an auditor's standard, and getting everyone able to do it, not only the enthusiasts. Three things follow from that, and they shape the rest of this report. Faster code generation moves the bottleneck to verification. Faster delivery without pipeline-generated evidence moves it to certification. And an initiative carried by one team cannot become an organisational capability.

Leadership's objective is equally clear. Current margins are healthy, and improving them is not the driver. The concern is future competitiveness, the pace at which new partners and jurisdictions can be onboarded, and more delivery capacity at roughly stable organisational scale. The ultimate outcome this engagement is measured on is therefore annual business output per team capacity, supported by the operational measures set out in Section 16. Cheaper delivery per partner is a welcome side effect, because it frees people for the jurisdictions waiting in the pipeline, it is not the goal.

The risk sits in the delivery model. Work is organised and funded per partner programme, which is a deliberate response to the market: adapting to each partner's acquiring set-up, settlement rules and reporting needs is how Northvale wins and keeps its partners, and dropping that adaptability is not an option. Per-partner analysis, compliance review and go-

live follow from it and will not go away. The rest does not follow from it. Scheme certification, regression rounds and environment set-up repeat because they are not automated. Connectors are rebuilt per partner because the unit of reuse is too coarse, not because the partner is different. The delivery model then closes the loop: partner commitments leave no room and no budget for the platform, quality and improvement work that would remove either, so gains stay local. Today that model delivers. The question is whether it can scale fast enough, demand already exceeds capacity, competitors are re-equipping with AI, and partners increasingly expect weeks rather than months to first transaction.

{the central conclusion}

Northvale's platform organisation is successful, growing and technically capable. The adaptability that wins its partners also means the delivery model leaves no owner and no funding for work that only needs doing once, and the four-weekly release window exists to compensate for verification that is still done by hand. That is now the constraint on growth. An AI-enabled Software Factory can increase delivery capacity, but only if it is combined with automated verification, clearer platform and partner boundaries, and explicit platform ownership.

{what this report is for}

Leadership asked for two things. Structure on how to make the transition happen, and capacity to implement the first pieces. Everything after this section answers one of those two.

The current-state value stream

The numbers below are the delivery baseline the factory is measured against. They describe a typical change for a typical partner, from first intake to steady-state operation, and they tell one story in three steps. The stage table gives the workshop's own estimates. The value-stream map decodes where the time goes. The delivery extract then confirms the same pattern in twelve months of measured data. Where a figure is measured, it comes from the extract; where it is an estimate, the text says so.

Stage	Duration today	What we heard	Source
Scoping & partner specification	~1.5–2 months	From intake to an agreed partner specification. Roughly three to four days of hands-on work, and the elapsed weeks are mostly waiting on partner responses and the compliance and risk review, not idle time. One programme carried six analysts.	Interview estimate
Development	~2 months	Implementation of the agreed scope (a connector plus the rails and core changes it needs, not a single change) across the gateway services and the ledger core.	Interview estimate
Certification & testing	~5 weeks	Regression is largely manual with partial end-to-end automation. Every core change is regression-tested before the release window, and scheme certification evidence is assembled by hand from test runs and screenshots.	Interview estimate + questionnaire
First production	4–5 months total	Intake to first production transaction for a typical partner change, including the wait for the next four-weekly core release window.	Interview estimate
New jurisdiction	12–15 months	From decision to a stable, steady-state operation in a new jurisdiction. A programme-level figure, unlike the per-change rows above. The most recent one took about fourteen months to first transaction.	Interview estimate

The delivery baseline. The near-term ambition is first production in two and a half to three months instead of four to five, resting on two assumptions: fewer iterations against an executable spec with unchanged partner response times, and regression coverage plus certification evidence built on the code actually being changed. One month is a longer-term aspiration, conditional on the factory maturing and the layer boundaries holding.

Value stream, one typical partner change

■ today ■ projected with the factory

Scoping call & brief

2d 1d

Partner spec & fit-gap (partner)

20d 8d

Compliance & risk review

10d 4d

Sign-off (x2)

3d

~7 weeks mostly on the partner and compliance side

Refinement & estimation

1d 0.5d

Backlog wait

8d

Implementation

20d 14d

Regression & scheme certification

15d 5d

Release-window wait (up to)

20d 1d

Deploy

1d 0.25d

3 sprints + window development, certification and deployment

The delivery value stream as mapped in the discovery workshop, in working days per step. Navy is the current duration, green is the projected duration once the factory is in place. Days are the workshop's own estimates, grouped into the partner-facing intake phase (about seven weeks, mostly waiting on the partner and on compliance) and the build-and-certify phase (three sprints plus the wait for the next release window). Projected reductions on partner-side steps come from fewer iterations against an executable spec and earlier validation, not from partners answering faster. Partner response time is outside Northvale's control.

Two things stand out. First, the largest single component is elapsed waiting time, split between partner and compliance feedback loops at the front and the release window at the back. Teams are not idle during those waits. They typically run several partner changes of this size in parallel, so the cost is long lead time per change, not unused capacity. Second, the hands-on work inside those waits is small. That is why the factory starts at specifications and verification: they are the cheapest places to buy back months.

The baseline, measured

412 delivered initiatives over twelve months, all values in days. Sizes are Northvale's own complexity rating. An initiative counts as delivered when its tracker status is Done and it has a confirmed delivery date. These are lead times, not deployment frequency, and Done is not a production deployment. Terms are defined in Appendix D.

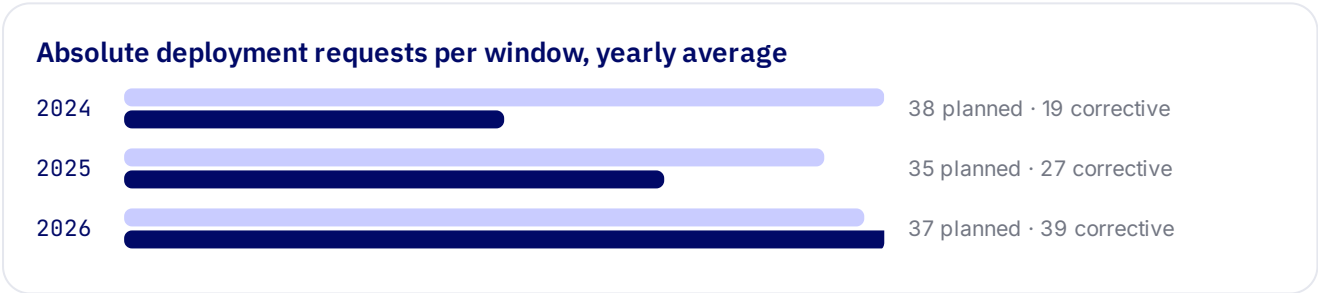
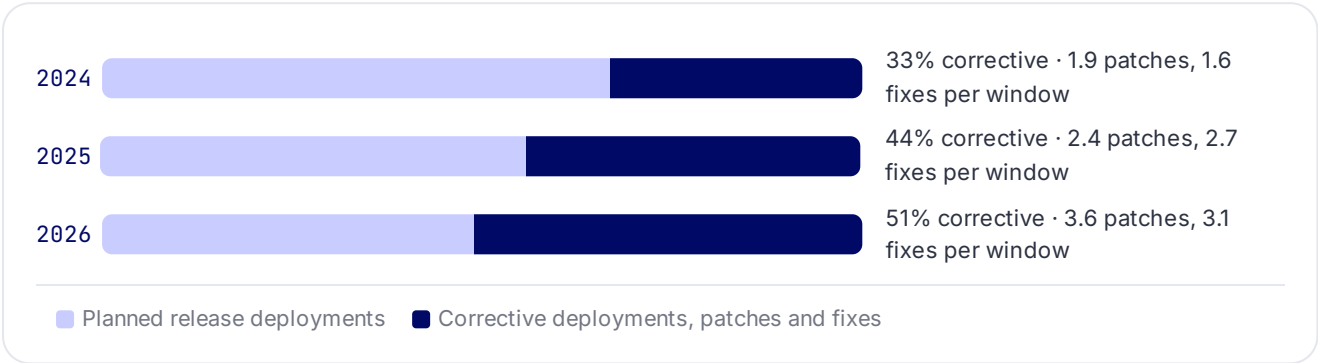
Size	Initiatives	Discovery to impl.	Impl. to done	Median total	P85 total
XS	61	22	19	44	98
S	158	29	33	63	141
M	127	33	47	84	176
L	52	48	71	121	212
XL	14	66	92	158	230

Segment medians do not sum to the total median. Two readings matter. Time splits roughly evenly between deciding what to build and building it, so accelerating implementation alone addresses half the problem. And P85 sits at a little

over twice the median at the common sizes, S to M, which is the signature of queueing rather than of work content. The extremes rest on small counts, fourteen initiatives at XL, so they are not quoted. The complexity rating is meant to exclude difficulty caused by the current technical landscape, so these durations are what it costs to deliver work of a given inherent complexity through the system as it stands, and the landscape cost is inside the numbers rather than inside the sizing.

The release window, measured

From Northvale's release records, every core release window from early 2024 to mid 2026. A four-weekly window carries on average about 95 tracker tickets and 40 updated deployable units into roughly 30 partner and jurisdiction environments. Corrective work is patches to acceptance plus fixes to production, measured as their share of all deployment requests.



How to read this. Percentages alone could hide a growing denominator, so the absolute numbers sit beside them. Planned deployments per window have stayed roughly flat while corrective deployments per window grew from 19 to 39, so each window needs about twice the repair events of two years ago regardless of what it carries. Throughput may well have increased. The efficiency of that throughput has measurably decreased. At 51 percent corrective, every added unit of deployment capacity buys about 49 cents of forward motion where 2024 bought 67.

Section 5 · Method

What we have done

The assessment ran over a two-day on-site discovery, a written questionnaire, a delivery-data extract and a leadership interview with the head of the platform organisation, and it drew on six sources.

- 01 **Structured questionnaire.** 24 responses covering engineering, analysis, testing, platform, product, compliance and management perspectives across the main delivery teams. It covered the flow of work from intake to release, system changeability, test and CI/CD maturity, current AI usage and reservations. The population is small, so questionnaire figures are treated as indicative rather than evidential.

02 Two days of on-site workshops. Group sessions captured goals, scope boundaries, risks and blockers from four groups across two rounds, and mapped the current-state value stream and the target flow with the full leadership and engineering group.

03 Fourteen recorded interview sessions, individual and in small groups, counted as sessions rather than people, with team leads, developers, testers, analysts, platform engineers, compliance, product and senior engineering.

04 Technical deep dive. We reviewed the platform architecture, including the ledger core and the gateway services around it, the connector and rails code for three partners, and the CI/CD pipelines across the delivery teams, together covering how changes are built, verified, certified and shipped today.

05 Delivery data. Initiative-level lead time and throughput for 412 delivered initiatives over twelve months, extracted under definitions agreed with Northvale, and release records for every core release window from early 2024 to mid 2026, with corrective deployments separated from planned ones.

06 Leadership interviews. A working session with the head of the platform organisation and the CTO on strategy, capacity, the state of the estate, the jurisdiction pipeline and the AI initiatives already under way.

What we found, in the order the work happens

This section follows one partner change through the organisation, in the order the work happens. The same failure repeats at every transition. Context is lost, work waits, and accountability changes hands. The scorecard at the end is the same findings in one view.

Intent

A change begins as a partner request. Analysts translate it into a specification, but the partner's own documentation is often out of date, settlement and reporting edge cases surface late, and there is no prototype for the partner to react to before code exists. Requirement quality depends on who brings the request and ranges from a single email to a fully formed specification. Only 3 of 24 respondents say requirements are very clear when development starts. There is a software development lifecycle but no consistent product development lifecycle.

The handover, first to compliance, then to engineering

Work cannot start until scope, edge cases, non-functional expectations and the compliance position are stable, and stabilising them is what most of this phase is spent on. It goes back and forth between analysts, compliance and engineering, with the partner intervening late. For the common initiative sizes, the median time from discovery to implementation is 29 to 33 days. The workshop map accounts for three to four days of hands-on work inside that window, so the rest is queueing and clarification.

Build, at two speeds

The gateway services are a strong direction and deploy on demand, roughly 25 times a day across 140 repositories. But significant flows stay coupled to the ledger core, shared settlement tables and partner-specific configuration, so teams cannot always build, test and release independently. A connector change that touches settlement waits for the core.

Dependence on scarce people

Delivery leans on a small group of engineers and analysts who hold the settlement and scheme knowledge, understand hidden dependencies and know how to navigate the organisation. When they are unavailable, work can stop. Depending on exceptional people to navigate undocumented complexity is not a capability. It is organisational risk.

Verification and certification, the largest single constraint

Northvale has substantial automated testing on the services estate, but not enough trustworthy, targeted, team-owned automation on the core. Every core change is regression-tested by hand before the window, test environments are shared and hard to reset, and the certification evidence a scheme audit needs is assembled from test runs and screenshots after the fact, roughly three weeks per change. Manual testing has become a source of perceived safety and accountability protection even where it is not formally mandatory. Engineering management named developer confidence when changing the core as the number one problem.

Release

The four-weekly window combines core changes from every team into one release and adds a one-week freeze, a manual regression round and staggered deployment into about 30 partner and

jurisdiction environments. Missing the window by a day costs the next slot. Independent production deployment already exists today for the services estate, so the window constraint is specifically the core path, not the estate as a whole. The window is not the only reason batching exists. Shared settlement interfaces are consumed without formal versioning, so a connector change and a core change often have to ship in the same slot whether or not they are otherwise related.

Operate

Once a connector is live it is maintained by the team that built it because nobody else can. Documentation lives in tickets, which describe what to do next but not how the flow works, and expertise on the ledger core is concentrated in a handful of people. Every change pays a knowledge-recovery tax before work starts.

And nobody owns the whole of it

Each function performs its part well. The end-to-end outcome belongs to no one.

The scorecard

For each area, the strength to build on, the constraint that limits it, the evidence, and what follows from it.

Area	Strength	Constraint	Evidence	Implication
Business & organisation	Growing, profitable, motivated teams, with leadership engaged and personally sponsoring the transformation.	Capacity, not demand, limits growth. Six jurisdictions live, three in the pipeline.	Leadership interview	Output per team is the lever. The factory is measured on it.
Intake & requirements	Experienced analysts, and a partner-onboarding team is forming to standardise intake.	Only 3 of 24 say requirements are very clear when development starts, and intake to an agreed partner spec runs 1.5 to 2 months.	Questionnaire + interview estimate	The largest front-end wait state and the factory's first target.
Platform vs partner ownership	Settlement, reconciliation and reporting run on largely shared code, and the newest jurisdiction was onboarded largely by configuration.	Shared modules are maintained where a team exists, but there is no mandate or funding for cross-partner assets beyond them. Connectors and rails packs diverge most.	Technical observation + interviews	Explicit layers plus lightweight governance for where functionality belongs.
Architecture & reuse	A shared core exists and is actively used. 31 live connectors share a common interface on paper.	Interview estimates suggest 40% to 70% of connector code is adapted or rewritten per partner. The MVP should establish a measured baseline by partner and module.	Interview estimate	The unit of reuse needs redefining. The three-layer model is the hypothesis to validate.
Development flow	Modern frameworks on the gateway services. A spec-first practice in pilot with human gates at spec, plan and merge.	Roughly two months of development per typical change. No consistent product development lifecycle across teams.	Interview estimate + technical observation	Standardise how work is specified, built and merged before scaling generation.

Area	Strength	Constraint	Evidence	Implication
Testing & certification	Gateway services sit behind quality gates with unit-test coverage above 85%. Scheme certification is well understood.	The ledger core has limited and uneven automated coverage. Every core change is regression-tested before the window, manually and with partial end-to-end automation, and certification evidence is assembled by hand.	Technical observation + questionnaire (17/24 patchy or minimal)	Automate verification and evidence before scaling generation, differentiated by risk.
Release	The services estate deploys on demand, roughly 25 times a day across 140 repositories.	The ledger core ships in a four-weekly window with a one-week freeze, and corrective work has grown from 33% to 51% of deployment requests since 2024.	Delivery extract (measured)	Take the low-risk class out of the window once verification is trusted.
Platform & infrastructure	Environments as code, centrally owned, consumed by every team. Already runs with factory-grade discipline.	At roughly four people, a capacity risk to validate before it owns the factory.	Technical observation + interviews	Natural factory owner, with named reinforcement (see MVP).
AI adoption & skills	24 of 24 use AI. Islands already show Stage 3 to 4 behaviours, and 11 of 24 would trust an agent with a small change today.	Real coordination exists (the spec-first pilot, a managed Copilot-to-Cursor migration, a prompt library). Missing is a central harness, consistent rules and skills, automatically enforced gates, an owner and measurement.	Questionnaire	Coordination is the current bottleneck. Shared foundations let the islands spread.
Security, compliance & governance	Strong tenant isolation and PCI discipline, with clear governance appetite in the discovery.	No agent identity, sandboxing or audit model yet. Rules for cardholder data, secrets and agents are undefined, and 9 of 24 name audit exposure as a reservation.	Technical observation + questionnaire	Build the security floor before agents scale (see Appendix A).

Assessment scorecard across ten areas. Evidence labels: delivery extract, interview estimate, questionnaire result, technical observation, external benchmark.

Six patterns that make delivery expensive, and the structural answer

The findings in Section 6 are one change's story. Read across all the evidence, they condense into six patterns, each observed in more than one place and named by more than one person. AI can already produce the change, but the system around it cannot cheaply define, contextualise, verify or certify the change. Where a finding matches a named pattern, we cite its code (for example AP-32 or TF-24). These codes come from the [re:cing transformation pattern library](#), part of Waves of Innovation, and every pattern cited in this report is described in Appendix B.

1 • Work starts unclear and comes back as rework [Stage 5](#)

Only 3 of 24 respondents say requirements are very clear when development starts. That rarely means restarting analysis or going back to the partner. Developers implement one part of the flow without the full context and lean on analysts and the compliance team for support along the way. Scoping runs roughly one and a half to two months from intake to an agreed partner specification, with three to four days of hands-on work inside weeks of mostly partner-side and compliance waiting, and one programme carried six analysts. Rework is driven by requirement misunderstanding, settlement and reporting edge cases that surface late and are not accounted for in the programme scope, and defects found in certification. Partners see demos roughly monthly, but formal validation still comes late, so misalignment is discovered late in delivery. On the ladder this is the Stage 5 bottleneck, and it is already binding at Stage 2.

2 • Correctness is certified by hand, and the release window compensates [Stage 6](#)

The picture is bimodal. Net-new code in the gateway services is in good shape, quality gates run on new commits, unit-test coverage exceeds 85%, and teams actively refactor into newer frameworks, which is worth saying plainly. The ledger core is the other half. It is stable, changes little and is routinely reused as-is, which is precisely what makes onboarding a partner on shared settlement cheap. The core itself is not the weakness. Its limited and uneven automated coverage is. The 17 of 24 who describe coverage as patchy or minimal are describing that gap. Because nobody can be sure what a core change breaks, every core change is batched into a four-weekly window with a week-long freeze and a manual regression round, and scheme certification evidence is assembled by hand from test runs and screenshots. The release records show the cost rising: corrective deployments have grown from a third of requests in 2024 to half in 2026 while planned deployments stayed flat. Leadership deliberately prioritised partner functionality over broad core modernisation, which is a defensible policy. What is missing is a sufficiently explicit, written, risk-based strategy, so the policy operates unevenly rather than by design. Where verification is still manual, agents amplify whatever verification exists, so faster generation there just fills the manual queue faster. The report's ask is not to build coverage everywhere, nor to prescribe a rewrite. It is that the core be given an explicit decision instead of a default. The options are to keep it frozen and wrap it behind stable interfaces, to extract from it incrementally with the strangler pattern, or to rewrite selected components with AI assistance, and choosing between them is a business decision the organisation has not yet made.

3 · Knowledge lives in people, not in reusable artefacts [Stage 4](#)

The ledger core is described as undocumented, existing documentation is outdated or unreadable for AI tools, and expertise on settlement and scheme rules is concentrated in a small group of people who have carried it across several partner programmes. Every change pays a knowledge-recovery tax before work starts. This is the Stage 4 context problem. It is also why past proofs of concept are so instructive. Where a connector was well documented, agents delivered a working change. Where it was not, humans reconstructed intent by hand first. The organisation has already run the experiment that proves docs-first sequencing.

4 · Connectors are rebuilt per partner because the unit of reuse is too coarse [Diagnosis](#)

Thirty-one connectors are live and they share a common interface on paper. In practice, interview estimates put 40% to 70% of connector code as adapted or rewritten per partner, and the rails that carry a jurisdiction's local payment methods and regulatory reporting are forked per partner rather than shared per jurisdiction. The usual reading is that this is inherent, because every acquirer and bank behaves differently. The more useful reading is that a single connector binary is trying to carry three different things at once: mechanics that never vary, rules that vary per jurisdiction, and partner-specific glue. The diagnosis chapter takes this apart.

5 · Local gains don't compound yet [Stage 2](#)

There is real, deliberate coordination under way. The spec-first pilot involves two teams with a fortnightly review, the Copilot-to-Cursor migration was managed with about fifteen people involved (15 of 24 respondents now use Cursor), a shared prompt library exists, an MCP integration for the ticket tracker is in use, and a ticket-to-merge-request pilot runs with one team. What is missing is the harness that would make these gains compound. Agent rules and skills are not consistent across teams, quality gates are enforced socially rather than automatically, and there is no owner, budget line or measurement of AI's effect. There is a structural reason. Programme leads have full autonomy over their teams' practices, so nothing can be enforced across programmes without going through the head of the platform organisation, and the programme-driven funding model means no single programme will pay for cross-cutting investment, the same trap that ended previous technical-debt initiatives. In pattern terms the risk is Accidental Transformation (AP-32), adoption outpacing the strategic harness, and it resolves by making the sanctioned path better than any workaround.

6 · Capacity is planned at one hundred percent and improvement is what gets cut [Funding](#)

Every interviewee described their team as fully utilised. Improvement ideas from engineers get pulled back into partner work at planning, and the previous test-automation initiative ended the same way. The organisation wants to invest in the platform, allocates a few percent to it, and is then surprised that the most advanced agentic set-up was built by one engineer in their own time. A factory built in the gaps of a full workload becomes one more island (Incremental Facade, AP-4).

The reservations respondents voice, reliability and hallucinations (10 of 24, the most named), compliance and audit exposure (9 of 24), and losing understanding of the codebase (7 of 24), are practitioner reservations. They match, almost word for word, what people at other organisations told us before our programme, and they point at exactly the gaps above. They are addressed by the same

fixes, and the second of them, audit exposure, is why the security floor in Appendix A is built before agents scale rather than after.

The structural answer: platform, rails and connectors

This assessment does not introduce an unfamiliar idea. It makes explicit a tension Northvale already recognises: partner flexibility is commercially essential, but unclear platform and partner boundaries limit reuse. Interview estimates suggest that 40% to 70% of connector code is adapted or rewritten per partner, and the MVP should establish a measured baseline by partner and module. The usual reading is that connector standardisation is structurally impossible here, because every acquirer, bank and marketplace behaves differently and partners buy precisely because the platform bends to them. The more useful reading is that the unit of reuse has been too coarse. A single connector cannot serve jurisdictions that share little regulatory logic, so it gets branched out per partner, and the jurisdiction rules go with it. The three-layer model gives a practical way to preserve that flexibility while increasing standardisation and ownership.

Decompose the what into three layers and two of them turn out to be standardisable after all. Mechanics that never vary, the ledger, settlement engine, message handling and the connector framework itself, are the real product, owned once and never forked. Rails packs are local payment methods, regulatory reporting and scheme variations captured as versioned assets per jurisdiction, built once for a jurisdiction and reused across every partner in it. Only the partner glue stays programme work, and even there the goal is to keep it thin and measured. So the clean dichotomy, standardise the how but never the what, becomes standardise the how everywhere and the what wherever variance is actually low.

This is not theory at Northvale, it is already happening. Settlement, reconciliation and reporting run on largely shared code, and the newest jurisdiction was onboarded largely by configuration with a small team. The rewrite problem is specific to connectors and rails, the historical growth area, where implementations genuinely diverged, which is exactly the layer where the unit of reuse was wrong. Partner behaviour has moved too. A few years ago partners pushed hard for deviation. Today, told that deviation costs more and takes longer, they mostly accept the standard. So the finding is not that standardisation is impossible. It failed where the unit of reuse was wrong, it shows promise where it was not, and partners now reward extending it.

Three kinds of repeated work

"Bespoke versus product" is the wrong axis for this question. It answers how many times a piece of work happens, and says nothing about what each time costs or how much of each instance is genuinely partner-specific. Separating the three gives a clearer view of what is fixed and what is not.

What repeats	Why	Fixed or addressable	What changes it
Per-partner analysis, compliance review, go-live	Each partner is a separate commercial, regulatory and contractual instance	The count is fixed. The cost per instance is not	Automation and standard onboarding models. Northvale already did this one layer down: environments as code reduced the cost of each environment without changing how often one is needed
Regression, certification evidence, environment set-up	The work is manual, so it is redone rather than re-run	Addressable in full, and independent of the delivery model	Automated verification and pipeline-generated evidence, factory Phases 1 and 2. 17 of 24 respondents describe coverage as patchy or minimal, and 9 of 24 are unsure how mature CI/CD is on their own repositories
Connectors and rails rebuilt per partner	A single connector cannot serve jurisdictions with little common regulatory logic, so anything that varies gets branched out	Addressable in large part, not in full	The three-layer model. A jurisdiction's local payment methods and reporting built once and reused across every partner in it is a rails pack, not bespoke work. Settlement and reconciliation already run on largely shared code; connectors diverge most

Row one is the floor and we are not asking Northvale to move it. Rows two and three are the gap between that floor and where delivery sits today, and that gap is what the factory and the layer boundaries close.

{platform, not product}

one platform, three layers, aligned ownership, shared Factory

Layer 1

Mechanics

the real product · owned centrally · never forked

Ledger and settlement engine, message handling, reconciliation, the connector framework, batch processing, the merchant portal component library. Same everywhere.

standardise
fully

Layer 2

Rails packs

one team owns a jurisdiction · versioned

Local payment methods, scheme variations, regulatory and tax reporting, licensing constraints. The logic differs, so give it a stable home, config and rule sets behind an explicit interface. A jurisdiction built once, reused across every partner in it.

versioned per
jurisdiction

Layer 3

Partner glue

programme work forever · the honest remainder

The partner's acquiring credentials, settlement schedule, custom reporting formats, one-off screens. This stays programme work, and that is fine.

keep thin,
measured

Layers are defined by where the boundaries sit, not by a rewrite. Functionality is classified during specification and placed provisionally in a layer behind an explicit interface. A second use triggers a mandatory review to promote, demote or split it, with AI-assisted extraction on that second programme's budget. The metric that says whether the boundaries hold is percent customisation per partner.

Structure follows ownership

Layers hold only if the organisation is shaped to hold them. Each layer needs two kinds of owner: an architect who owns its architecture (boundaries, interfaces and how the layer is allowed to evolve) and a stable team that owns its product delivery lifecycle (roadmap, quality, operation and outcomes). This is the Inverse Conway Manoeuvre, shaping teams so the target architecture is the path of least resistance rather than letting programme structure dictate the code (Establish Team Topologies, TF-31).

The proposed team topology

Boundary	Architecture owner	Delivery ownership
Layer 1 · Mechanics	Architects	Core platform team, owns the PDLC from day one.
Layer 2 · Rails packs	Architects	Stable rails-pack team, owns and versions all packs centrally.
Layer 3 · Partner glue	Architect sign-off on interfaces	Partner delivery team, it is the programme.
Software Factory	Architects + platform lead	Platform engineering team.
Boundary governance	Architects	Classification, interfaces, exceptions and movement between layers.

Northvale's architects own the architecture of the layers, what belongs where, the interfaces between them, and how each layer is allowed to change. The stable teams own delivery within those boundaries. Mechanics are owned centrally from day one. Rails packs are created at programme level and then promoted into a stable team that owns and versions all of them centrally, rather than being forked per partner. Partner glue never leaves the programme. One question stays open: whether a single platform team owns the delivery lifecycle of layers 1 and 2, or two stable teams split it. One team keeps the two tightly coordinated but can become a bottleneck as jurisdictions grow, two teams scale better but need a clear interface between them. Our provisional view is to start as one team and split out a dedicated rails-pack team once the number of jurisdictions makes it a bottleneck, and this is carried as a decision in Section 12.

Placement is a lifecycle, not a one-time bet

Some functionality clearly belongs in mechanics or a rails pack from the start, and some only reveals itself as reusable later, so placement is a lightweight, repeatable process that also transfers ownership. The architect is the decision-maker at classification and at every promotion.

1. An architect classifies the functionality during intent and specification: mechanics, rails packs, or partner glue.
2. It is built in the programme, behind an explicit interface the architect approves, whatever its target layer.
3. Place by layer: partner glue stays programme-owned, mechanics that are clearly core are handed to the platform team on first release, and a rails pack stays programme-owned until it proves reusable.
4. Second use is the promotion trigger for a rails pack: an architect-led review promotes it into the stable rails-pack team, which takes over its lifecycle.
5. Promote, demote or split with AI-assisted extraction and refactoring, and ownership transfers with the code.
6. Record the decision and update specifications and tests.

The system assumes some initial placements will be wrong, so moving functionality and its ownership between layers has to be routine, safe and backed by automated tests.

This topology is an initial suggestion. A concrete team structure, including whether layers 1 and 2 share one owning team or split into two, should be defined and validated as part of Phases 1 and 2, on evidence from the pilot.

{the optimisation goal}

Maximise reuse while preserving necessary partner flexibility.

Not "rewrite partner functionality faster." Boundary metrics: percentage of partner-specific functionality, reused versus newly implemented components, duplicate capabilities across programmes, time and cost to promote functionality into a reusable layer, changes that cross layer boundaries, and maintenance effort by layer.

{the prize}

Product margins on bespoke work. The unit is no longer the product, it is the delivered change, and every change produced through the factory reuses the same specs, context, pipelines, tests, rules and agents, and leaves those assets better than it found them, so the next change is cheaper than the last. A second asset compounds beside it, the rails pack, so the durable, sellable thing becomes the spec, the domain model and the certified test suite, not the code.

It delivers exactly what productising the platform was supposed to deliver, the unit economics. The reason organisations chase a standard product in the first place is the margin arithmetic behind it: build once, sell many times, so the marginal cost of the next sale falls and margins grow with volume. In a services model, revenue and cost grow together, more partners means proportionally more people, and margins stay flat. A standardised way of developing moves Northvale toward the product-style curve without giving up customisation. Reuse, automation and AI compound and progressively weaken the link between cost and partner volume, so cost per delivered unit falls as volume grows, while every unit remains as partner-specific as the partner requires. Northvale gets paid for customisation and produces it at declining marginal cost. That combination, product margins on bespoke work, is the actual prize.

path A

AI applied to the current system

- Faster local coding
- Same handovers, same release window
- More partner-specific code generated
- Certification becomes the bottleneck
- Costs still grow broadly with partner volume

path B

AI combined with architectural and organisational change

- Faster intent clarification
- Reusable mechanics and rails packs
- Automated verification and certification evidence
- Lower partner-glue proportion
- AI-assisted extraction and modernisation
- Delivery cost per additional partner declines

Two transformation paths. Applying AI to the delivery system as it stands speeds up local coding and then stalls at certification, with cost still tracking partner volume. Combining AI with the layer boundaries and team topology lets reuse and automation compound, so the cost of each additional partner declines. The productivity multipliers often quoted for AI are illustrative scenarios, not forecasts.

The AI Adoption Ladder

We assess organisations against the re:cinq AI Adoption Ladder, a nine-stage model (Stage 0 to Stage 8) grouped into four phases. Each stage describes who does the work, where knowledge lives, and how correctness is confirmed, and each stage has one bottleneck that blocks the way up. The logic behind the ladder is Goldratt's Theory of Constraints, the same idea that has guided delivery improvement for forty years. The throughput of a system is set by its biggest bottleneck. Fixing anything else just piles work up in front of the real constraint.

Each stage reveals the next constraint, and progress comes from solving them one by one. The pill on each stage marks the key constraint that emerges there.

Discovery Getting AI into the company. Access, then coordination.	0	Curiosity No formal adoption, personal accounts	Org permission
	1	First Licences A 5–10% pilot group gets access	Access + no feedback signal
	2	Wide Individual Adoption NORTHVALE Most use AI, uncoordinated	Coordination
Discipline Making AI work for the team. Review, context, security, ownership, ops.	3	Structured Adoption Standards enforced in tooling + CI	Review at AI speed
	4	Agentic Team (depth) Multi-agent and sandboxed, context as code	Context + security + ownership
	5	Operational AI (breadth) AI across CI/CD, IaC, SRE, product	Scale + product flow
Factory Making AI work at scale. Validation, governance, the shape that feeds it.	6	Lights-On Factory TARGET Agents write the majority of code	Machine-speed validation
	7	Dark Factory Humans specify, agents implement	The frontier, emerging
Frontier Undefined. Few teams are here and practices are still forming.	8	Beyond Not yet known	Undefined

Adapted from the re:cinq AI Adoption Ladder whitepaper. Most teams sit around Stage 2. Northvale sits at Stage 2, with team-level islands at Stages 3 to 4, and the target is Stage 6.

AI applied to code generation is the clearest recent example. If writing code was your bottleneck, throughput improves. If it wasn't, you made one part of the process faster and moved the queue. The bottleneck never disappears, it moves, and climbing the ladder means finding and fixing the current

one, in order. Organisations that try to skip stages inherit the unsolved constraints of every stage they skipped.

The ladder applies per team, per codebase. No company sits on a single rung. The organisation is read as the sum of those positions, and it only climbs as fast as its codebases align.

0-2 Discovery. Getting AI into the company at all.

Stage 0 is curiosity, individuals trying tools with no formal adoption. Stage 1 is first licences, a pilot group of 5 to 10% gets access and the rest of the organisation has no reliable signal about whether it works. Stage 2 is wide individual adoption. Most people use AI, everyone uses it their own way, with different tools, prompts and conventions on the same codebase. Gains are real but personal, knowledge lives in individuals' heads and prompt files. The bottleneck at Stage 2 is coordination. The dominant work in leaving Discovery is education and first shared conventions. Behaviour change here takes weeks of practice on real work, not one-day workshops.

3-5 Discipline. Making AI work reliably for the team.

Stage 3 is structured adoption. Agents handle small, well-scoped changes under human review, and standards are enforced mechanically through tooling, linting and CI, because when many developers each run their own agent with their own context file, output converges on the model's defaults rather than on the team's standards. The bottleneck at Stage 3 is review, since output now arrives faster than senior engineers can carefully read it.

Stage 4 is the agentic team. Several agents per developer is normal, running sandboxed, and context becomes shared infrastructure: versioned rules, skills and machine-readable documentation containing only what is not the model's default, the invariants, exceptions and conventions specific to this codebase. This is where the security model needs a real answer, because an agent must not run with a developer's full permissions, data access and network reach, and in a PCI environment that answer has to be auditable. The bottleneck is context quality, security and ownership.

Stage 5 is operational AI, breadth. Agents execute well-specified tasks end to end, most of the team works this way, beyond the early enthusiasts, and AI spreads beyond the IDE into CI/CD, infrastructure as code and review automation. The bottleneck shifts away from engineers toward the people around them, analysts, product, compliance, who have not yet adjusted how they hand work over. A vague instruction makes the agent decide on your behalf and report success anyway, and most teams at this stage have a ticket and a hope, not a spec an agent can execute well. Specification quality is what gates the way into the Factory.

6-7 Factory. Making AI work at scale.

Stage 6 is the lights-on factory. Agents produce most of the work in units the organisation can review (today the pull request is that unit) and humans sit at deliberate validation gates. Standards, tests and acceptance criteria are enforced mechanically, because anything that depends on human discipline does not survive AI-speed output volume. The bottleneck is validation at machine speed, since agents cannot reliably validate their own output, so the validation pipeline sets the pace for everything. Extending the factory into operations, deployment and maintenance belongs to this phase too, with an approval model closer to production database access than to coding assistance, because ops agents take actions rather than propose code. A factory is to AI-native delivery what an internal developer platform was to cloud native: central infrastructure that every delivery unit consumes instead of rebuilding.

Stage 7 is the dark factory. Humans specify, agents implement, and validation carries the weight review used to carry. Public references exist and they are evidence the rung exists, not a recommendation. For most organisations today, including a regulated payments platform, it is not a sensible target.

8+ Frontier.

Unknown. No team we know has gone past the dark factory.

Two things run across the whole ladder rather than sitting on one stage. Measurement, because activity metrics like PR count and token usage stay green while the system rots, so the ladder is tracked with throughput metrics, lead time, change failure rate, time to restore, and the value stream from spec to production. And the human role, which moves as you climb, from writing the work, to reviewing the work, to defining and validating intent.

Where Northvale stands

The organisation as a whole is at **Stage 2, wide individual adoption**. Everyone uses AI tools and gains are real, but they are personal, uncoordinated and unmeasured. In our field data this is the most common position, roughly a third of organisations sit here, and fewer than a quarter have made it past Stage 3. Northvale is not behind. It is at the plateau where most of the industry is stuck, and the ones that get unstuck first take the advantage.

The eight-question diagnostic

Each statement scores true only if it holds across the organisation today. Islands that score higher are recorded in the stage map below. Scored from the questionnaire and workshop data.

1	Most developers use AI daily, not just the eager few.	True
2	Written, tooling-enforced rules for how AI-generated code is reviewed.	Partial, not unified
3	Agentic tools have scoped credentials and run sandboxed.	Partial, one team
4	A structured context file per repository, holding only non-default rules.	Partial, unmaintained
5	CI fast and trusted enough to be a real-time feedback loop.	No
6	Test suite catches regressions reliably, so AI code adds no hidden review burden.	No
7	Delivery throughput measured (lead time, change failure rate, time to restore).	Partial, initiative level only
8	Agents merge real code behind validation pipelines with written rules on what ships automatically.	No

1 / 8

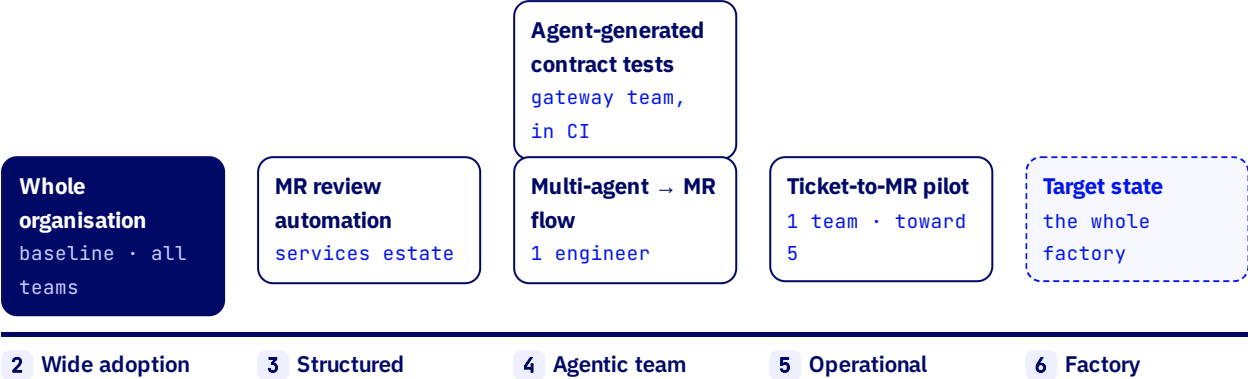
Discovery, consistent with Stage 2. Four partials, three outright no. No question passes and only three fail outright, which is what a stalled Stage 2 looks like from the inside. Re-score quarterly on this same instrument.

The islands

Assessed per codebase, as the ladder requires, the picture is uneven in an encouraging way. Agent-generated contract tests run in CI for the gateway team, automated merge-request review runs across the services estate, one engineer runs several agents in parallel through a rules-and-skills implementation-to-merge-request flow, and a ticket-to-merge-request pilot runs in one team. These are Stage 3 to 4 behaviours, individual teams already working the way Discipline demands. What is missing is everything that would let them spread: the Stage 3 foundations (standards enforced in tooling, structured education, redesigned review) and the Stage 4 foundations (versioned context infrastructure, machine-readable documentation, a security and ownership model). No owner, no measurement. That missing middle is exactly why the islands stay islands.

{per-team stage map}

each chip is one team or practice



The missing middle. The Stage 3–4 foundations no team has built for everyone.

Every delivery team sits at the Stage 2 baseline, while four team-level islands run at Stage 3 to 4 behaviours, one reaching toward Stage 5. Nobody has built the shared Stage 3–4 foundations between them, which is why the islands stay islands.

Maturity is uneven, and the average hides the point

A single organisation-wide score would mislead. Assessed by group, the picture is uneven in an encouraging way.

Group	Observed practice	Indicative stage
Gateway team	Agent-generated contract tests running in CI, a structured context file per repository, spec-first working practice in pilot.	4
One advanced engineer	Several agents in parallel across repositories through a rules-and-skills implementation-to-merge-request flow, with a personal skill library.	4
Services estate	Automated first-pass merge-request review, quality gates on new commits, on-demand deployment.	3
Ticket-to-MR pilot team	Tickets dispatched to an agent that opens a merge request, human review on every one.	3, reaching toward 5
Analysts and compliance	Active AI use for drafting specifications and reviewing partner documentation, no shared templates.	2 to 3
The middle group	Willing, using assistants for individual tasks, with no shared practice or guardrails.	1 to 2
Organisation-wide foundations	No enforced standards for AI-generated code, no versioned context infrastructure, no ownership or security model for agents, and no measurement.	2

The islands are real and they are ahead of most organisations. What is missing is everything that would let them spread, which is the organisation-wide foundations of Stages 3 and 4. The diagnostic above is the instrument, and it is re-scored quarterly.

The infrastructure layer

One layer of Northvale already runs with Stage 6 discipline. Environments are as-code, centrally owned and consumed by every team, and the services estate deploys to production roughly 25 times a day across 140 repositories without a window or a ticket. This is the internal proof that the factory model works at Northvale. The task is to build the same discipline for the stages in between, in order, rather than jumping from Stage 2 to Stage 6.

The findings map onto ladder bottlenecks

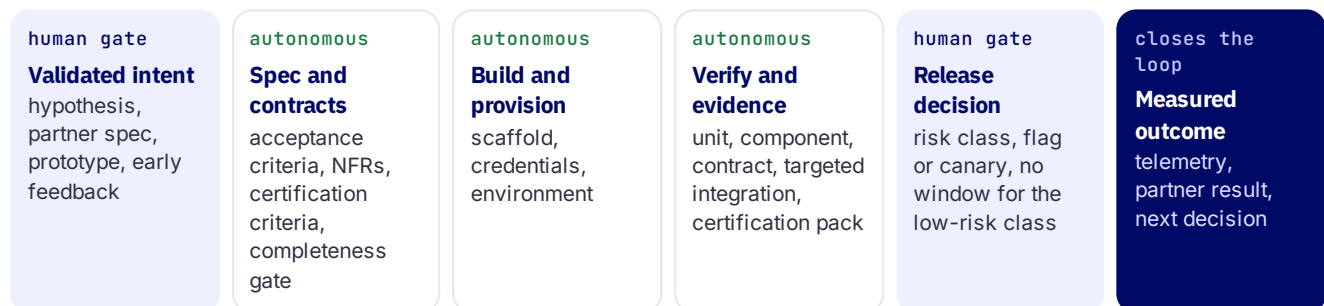
Requirements clarity is the Stage 5 specification problem, the handover from analysts, compliance and partners is not yet shaped for agent-speed work. Manual verification and hand-assembled certification evidence are the Stage 6 validation bottleneck arriving early, faster generation just fills the manual queue and the release window faster. The undocumented ledger core is the Stage 4 context problem, every change pays a knowledge-recovery tax before work starts. None of these is unusual. What the ladder adds is the order in which they must be fixed.

The target AI-enabled SDLC

The committed target is Stage 6: agents produce most of the work in units the organisation can review (today, the pull request), and humans sit at deliberate intent, merge, risk and release gates. Standards, tests, acceptance criteria and certification evidence are enforced and produced mechanically, because anything that depends on human discipline does not survive AI-speed output volume. Approval gates are the design, and they answer the practitioner reservations about reliability and audit exposure directly: the reviewer at the merge boundary sees the spec, the diff, the validation report, an independent review and the full audit trail, and the same trail is what an auditor sees. This is Human-in-the-Loop Validation (AN-14) applied deliberately.

```
{on autonomy}
```

Greater autonomy than Stage 6 is a longer-term option, conditional on evidence, risk and technological maturity. It is not the committed target, and it is not permanently ruled out. The reference platform described in Appendix A ships a full-autonomy mode that is off by default behind two independent switches, one per repository and one per cluster; enabling it requires deliberate, secondary approvals by an administrator and a code owner through a labelled change. The capability existing and the capability being on are separate decisions.



The target value stream. The same stages as today, two permanent human gates in place of six handovers, and a closing loop where today there is an open end. Between the gates the number of human checkpoints depends on the risk class of the change, and it starts higher than it ends. Each checkpoint becomes mechanical when the signals behind it can be trusted, and not before (Delayed Automation, TF-20). Checks stay human permanently where the risk or the regulation warrants it.

How work flows through the factory

The factory standardises the how everywhere: specifications, context, pipelines, verification, evidence and agents are the same across teams. It is built in two stages, in line with how the constraint moves.

Stage one, specifications and verification first. The specification path has to be built in any case, and it delivers the most value fastest: shorter lead time from intake to approved specification, less rework, and clean, consistent documentation. Beside it, a first automated regression suite on the pilot scope, with certification criteria attached to the spec so the evidence a scheme audit needs is produced by the pipeline rather than assembled afterwards. It also lets analysts and compliance start thinking ahead of new functionality instead of chasing clarifications. Developers gain immediately, from clear specs instead of vague tickets and from documentation that is finally consistent. In this

stage, code generation improves using existing, semi-structured AI-assisted coding, which is already proven inside Northvale's islands.

Stage two, generation and guardrails. With specs, context and trusted verification in place, invest in skills and harnesses for agents, scaled code generation, agentic test generation wired into CI, and quality, security and performance guardrails as fixed pipeline stages. Verification is automated before generation is scaled, because the Stage 6 constraint is validation at machine speed, and faster generation against a manual queue just fills the queue faster (Faros telemetry across 10,000 developers shows 98% more pull requests merged with heavy AI use while delivery outcomes stayed flat, with review time up 91%). This is also where the low-risk change class leaves the release window.

Partner validation moves before development

Generated specs plus quickly built, presentable prototypes let Northvale agree intent with the partner before code is written. This addresses the largest front-end wait states, clarification and partner feedback, at their source, and turns late-feedback rework into early sign-off. It is Intent-Driven Development (AN-10) in practice.

The release window becomes a risk class, not a calendar

For software that can deploy independently, the four-weekly window is an operating-model defect rather than an immutable constraint, and treating it as immutable is what keeps confidence manual. Once the pilot's regression suite and certification evidence are trusted enough to gate a release, low-risk changes on the pilot component deploy behind flags and canaries with automated evidence, and the window stays temporarily for tightly coupled core flows only. High-risk and scheme-impacting changes keep every control they have today.

The enablers this depends on

Four capabilities have to exist for the flow above to run. They are enablers of the end-to-end objective rather than objectives in their own right, and each one gets built inside the pilot instead of ahead of it.

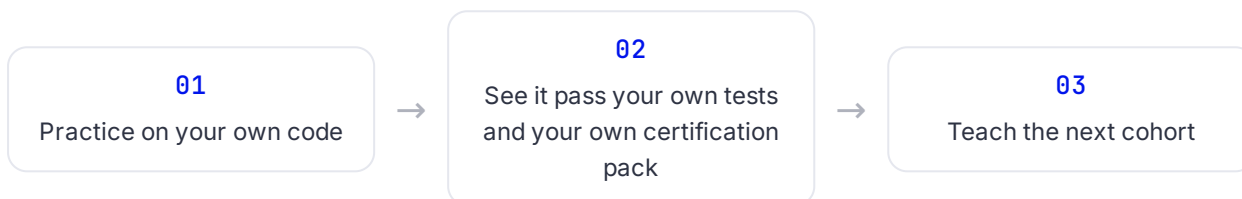
Enabler	What it is
Executable specifications	The partner change described in one structured form that analysts, compliance and agents all read, with acceptance criteria, non-functional requirements and certification criteria attached, and a clickable prototype generated from it so the partner reacts before anything is built (Intent-Driven Development, AN-10).
Shared context	One context service for the whole organisation, fed on every merge and continuously from deployments, with facts carrying validity and decay so a stale status cannot masquerade as current. Its first content is the living specification of the pilot connector and the core interfaces it touches (Retrieval-Augmented Generation, AN-9).
Trusted verification and evidence	Contract tests between connectors and the core, a regression suite on the pilot scope that runs on every merge, and the certification pack produced by the pipeline as a by-product, so the evidence exists before anyone asks for it.
A Golden Path, not a queue	Approved model access, repository-level engineering and testing instructions, standard specs and contracts, CI/CD templates and quality gates, security and compliance controls, ephemeral environments, resettable test data, flags and canaries, audit trails and cost controls. Teams keep fast local feedback, and the platform's job is to make the standard path the easiest one rather than the only one (Minimize Cognitive Load, TF-45; Enable Self-Service, TF-47).

How adoption spreads

Migration onto the factory is gradual and evidence-led, teams join when the factory demonstrably beats their current way of working, following Platform as a Product (TF-24) and Gradual Onboarding (TF-32). Once a team migrates, the factory's standards, controls and paved road become the default for that team, with lightweight, governed exceptions. Voluntary entry, disciplined residence.

Mindset change is budgeted as a deliverable with its own time and owner, because the constraint is how people define work, hand it over and trust what an agent produces. The method is specific: conviction is earned on a person's own code across weeks of practice, then spread by the people who earned it. The existing engineering guild meeting becomes the venue, upgraded rather than replaced. The human role moves up, from writing the work to reviewing it to defining and validating intent, and it should be named as the promotion it is.

`{conviction loop}`



The loop repeats per cohort. Each round of taught engineers becomes the teachers of the next, so conviction compounds instead of resetting with every workshop.

Ways of working, later

Some coordination ceremonies may become redundant once the factory reliably supplies the information or control they exist to provide. Task state visible in the pipeline can replace status meetings, pipeline-generated evidence can replace the pre-window certification review, and a measured cost baseline changes what estimation is for. No ceremony is retired before it is reviewed against one question: does the factory now provide what this meeting exists to carry? The retrospective and the engineering guild are standing exceptions, and where partner contracts expect a release calendar, the external cadence keeps its slot.

Section 11 · Quality

The quality operating model

Build trustworthy quality signals first, connect them into agentic loops second, and grant higher autonomy only when the evidence earns it.

Northvale should not begin with a broadly autonomous pipeline while regression on the core is manual, test environments are shared and certification evidence is assembled by hand. The cost of the current model is measurable and rising: the release records in Section 4 show corrective work growing from 33 percent of deployment requests in 2024 to 51 percent in 2026. The pre-mortem's own fear, generated code that nobody can maintain and an audit finding nobody can explain because an agent produced it, is the outcome of skipping this section.

level 1

Agentic development loop

Fast feedback close to the code through unit, component and contract tests generated with the change, a simplicity gate on generated code, and a review agent with fresh context that never sees the generation session. Engineers own the gates nearest their code.

level 2

Agentic quality-control loop

Deployed validation through integration, end-to-end, security, performance and observability signals, contract compatibility checks against the core, and the scheme certification criteria as executable acceptance tests that produce the evidence pack.

level 3, later

Agentic software delivery loop

An orchestration layer that turns reliable findings into work for development agents. Only once levels 1 and 2 can be trusted, and only for the low-risk change class.

Levels 1 and 2 connect to the existing QA and compliance programme rather than running beside it, so the programme's gates become the levels' trusted signals. Two things in Northvale's internal quality strategy, shared with us during discovery, deserve protecting as the levels get built. Test environments and test data are strategic capabilities rather than project overhead, and test generation should start with narrow agents tied to a specific gate rather than one general QA agent. The one warning is the one every team learns once: an agent that writes the code and the test that checks it will make the test pass. Validation has to be independent of generation. Who owns which gate is a decision in Section 12.

What leadership commits to, and what waits

Three commitments have to be made before the MVP starts. Everything else can be decided inside the first weeks, with the pilot in front of it, and is better decided there than in advance.

The three commitments

End-to-end ownership

The pilot team is accountable for the outcome from intake to production, and four roles come as a set, proposed as functions for leadership to fill rather than named here. An executive sponsor who carries the capacity decision and the public message (Executive Commitment, TF-11). An objective owner accountable for total lead time and delivered value across the journey. A transformation champion who drives it day to day and owns adoption (Appoint Transformation Champion, TF-12). And a dedicated cross-functional core team for the first slice (Form the Core Team, TF-13), which is the pilot team in Section 13. Without this, none of the rest holds.

Capacity, from named deferrals

The programme gets its capacity by taking it from something named. Leadership reduces or defers specific partner commitments, says which ones, reinforces the platform engineering team with one or two engineers seconded from the gateway team, and decides before more tooling gets built. The alternative is absorbing the work into current workloads, and that is the failure mode the organisation has already seen.

One team, one change, one complete loop

A twelve-week slice, run as two six-week increments with a checkpoint between them, measured at both ends and funded to a decision point rather than to a destination (Staged Funding, TF-23). The team is entitled to change what stands in its way, including the release window for its change class, the certification review and the intake chain. Section 13 is what that commits to.

Decided in the first weeks

Seven decisions determine whether the factory compounds or dies the way previous initiatives did. They are Northvale's to make, during the MVP, with evidence from the pilot. For each, the table shows the main options and the trade-off, with re:cinq's provisional view where we have one.

Decision	Options and trade-offs	Provisional view
Ownership of the ledger core	Keep frozen and wrap behind stable interfaces (lowest risk, caps reuse). Strangler-extract incrementally (steady progress, needs protected capacity). AI-assisted selective rewrite (fastest, highest risk in a regulated ledger). A written, risk-based core strategy is the deliverable either way.	Decide with MVP evidence. Strangler-extraction is the default hypothesis.

Decision	Options and trade-offs	Provisional view
Authority to protect core and platform capacity	A fixed capacity ring-fence per team versus an escalation rule when programmes claim platform people. Ring-fencing is predictable but rigid. Escalation is flexible but erodes under pressure.	Ring-fence a small fixed share, reviewed quarterly.
Factory ownership	Platform engineering (most mature, but ~4 people). A new platform team (clean mandate, slow to form). The spec-first pilot's authors (closest to the delivery loop, furthest from platform practice).	Platform engineering, reinforced by secondment from the gateway team.
Core and rails-pack team structure	One platform team owns the delivery lifecycle of layers 1 and 2 (tight coordination, risk of bottleneck) versus two stable teams, one per layer (scales across jurisdictions, needs a clear interface). Rails packs are owned and versioned centrally either way, and architects own the layer architecture in both.	Start as one team, and split out a rails-pack team when jurisdictions make it a bottleneck. Confirm in Phases 1 and 2.
The release window	Keep the window for everything (no change, confidence stays manual). Remove it for a defined low-risk class once the pilot's evidence is trusted (controlled, needs the risk classification agreed with compliance). Remove it entirely (premature while the core's coverage is uneven).	Remove it for the low-risk class in the MVP, keep it for coupled core flows until Phase 2.
Funding for cross-programme capabilities	Staged central funding released against demonstrated milestones (Staged Funding, TF-23) versus extraction riding the second programme's budget (rule of two). These combine rather than compete.	Rule of two for extraction, and staged central funding for the factory itself.
Protected improvement and technical-debt capacity	Explicit percentage of capacity per team versus improvement time claimed from programme slack. Slack does not exist in this organisation; if the time is not explicit, it is not real.	Explicit, small, non-negotiable allocation from Phase 2.

The report shows alternatives and trade-offs. Northvale decides. The MVP exit review is the natural decision point, with one exception. Whether the three-layer platform architecture is inside MVP scope must be decided before the MVP starts.

Twelve weeks, one team, one change

These twelve weeks build the Specification and Verification Factory, the first capability of the wider Software Factory and the shared platform of context, pipelines and agents that every later capability runs on. Its primary outcome is an approved, executable specification with acceptance and certification criteria attached, backed by a validated prototype, living documentation, and a first automated regression suite trusted enough to gate a release. To prove those specifications are usable, one bounded change is then taken through Northvale's existing SDLC and deployed outside the release window, showing that the specification can drive both implementation and verification without yet industrialising agentic coding, which belongs to Phase 2. It is deliberately only one of the factory's capabilities, chosen first because it is the highest-leverage, lowest-risk part of the climb. It runs on one pilot connector team with a small crew of motivated volunteers, inside the already-approved tool stack, building on the spec-first pilot and the gateway team's contract-test work. The first job is to win hearts and minds with something simple but immediately valuable. Even if Northvale stopped after the MVP, it would still own the intake and verification machinery, the two most valuable pieces.

We recommend a connector team working on a modern gateway component as the pilot, not a settled choice, and the criteria matter more than the name. The team needs a change class that can ship independently, an analyst able to state a measurable partner hypothesis, a certification scope small enough to automate in twelve weeks, and enough slack to absorb the work. Leadership confirms the team in week one.

The twelve weeks, concretely

1. Confirm the pilot team and its scope with leadership in week one.
2. Establish baselines: intake time, handovers, documentation quality, cost per change, certification effort per change and percent customisation for what goes through the factory.
3. Generate living specifications from the existing codebase of the pilot connector, its rails and the core interfaces it touches, plus the existing analytical documentation, kept in sync on every merge.
4. Add meeting capture as a third context source beside code and analytical documentation. Partner specification sessions and internal reviews feed the living specs, with explicit rules for what is not recorded and who can access transcripts, and with cardholder data never in scope.
5. Validate generated specs continuously. A weekly spec-drift job compares specs against the actual code, so human review targets drift instead of full re-reads.
6. Stand up the first automated regression suite for the pilot scope, extending the agent-generated contract tests already proven in the gateway team, and attach the scheme certification criteria to the spec so the pipeline produces the evidence pack.
7. Build the prototype intake loop, so intent is agreed with the partner against a presentable, clickable version of the change before any production code is written.
8. Make the specs a queryable knowledge service, with two consumer groups live: the spec-first pilot's skills and the pilot team's analysts.

9. Take one bounded change through Northvale's existing SDLC from the approved specification, verified by the new suite, and deploy it behind a flag outside the release window, to prove the specification can drive implementation and the evidence can drive release. Industrialising agentic coding belongs to Phase 2.
10. Measure the baseline of cost per change and percent customisation, plus the throughput metrics (lead time, change failure rate, time to restore) that every later phase is judged against.
11. Draft the team structure for the layers (core, rails and factory ownership, with architects owning the layer architecture), decide factory ownership, governance and the next migration cohort (see Section 12).

The five workstreams, each with an owner and an exit criterion

These are five lenses on one team's work, not five separately staffed streams. The MVP runs on a single dedicated cross-functional team drawn from the pilot team, platform engineering, quality and compliance, roughly ten people, with re:cinq's contribution set out below.

Workstream	First action	Owner	Exit criterion
Intent and specification	Intake checklist and completeness gate on the pilot change, with acceptance, non-functional and certification criteria made explicit.	Analyst lead, pilot team	One bounded change travels from captured intent to production through the gates.
Verification and evidence	Automated regression on the pilot component, resettable test data, certification pack generated by the pipeline.	Quality lead + compliance	A measured statement of what the pilot suite actually catches, trusted enough to gate a release, and an evidence pack accepted by compliance.
Release path	Risk classification, flags, canary and rollback for the pilot change class.	Platform engineering	The pilot deploys to production without the window and without a manual ticket.
Knowledge and context	Living specs and a context file for the pilot repositories, retrieval available to agents and humans.	Architecture	The pilot team answers architecture and history questions from the system, and ramp time is measurably lower.
Measurement	Week-one baseline on the agreed instrument.	Head of platform organisation	The MVP exit review is decided on measured evidence, not sentiment.

Week one is measurement

Confirm the pilot team and the change. Baseline the measures the programme is judged on: lead time, waiting against active time, handovers and clarification cycles, time awaiting compliance, certification and release, manual test and evidence effort, escaped defects, new-engineer time to a production change. What the tracker cannot extract gets measured by hand, through sampling, shadowing and ticket-history counts. Agree it before anything gets built.

Week twelve is a decision, not a launch

The MVP ends in one of three answers, decided on the measured result, and funding for the next phase is released against it rather than committed up front (Staged Funding, TF-23). Scale it to the next teams. Adjust the approach and run a second slice. Or stop, having spent twelve weeks and learned what does not work here, which is a legitimate outcome and much cheaper than discovering it in month eighteen.

What re:cinq contributes

- A technical lead for the factory build and the extension of the spec-first pilot.
- Organisational and change leadership for the operating-model decisions, baselining and the adoption programme.
- Two hands-on engineers embedded with the pilot team for the duration of the MVP, with access to Northvale's code and tools under Northvale's own access controls.

This twelve-week outline is input to a separate MVP definition document, agreed with Northvale before the MVP starts. Duration and cost come as a separate proposal, scoped against this plan.

Ownership and staffing during the MVP

The provisional factory owner is the platform engineering team: the factory is a platform, and on the practices it needs they are the most mature team in the organisation. At roughly four people they are a capacity risk to validate, and likely too thin to take this on unaided, so reinforcement is a named part of the plan: second one or two engineers from the gateway team for the MVP period, with a permanent staffing decision at the MVP exit. Ownership is split rather than piled on one team. Platform engineering owns the factory as a platform, the spec-first pilot's authors keep the delivery loop that runs on it, and ownership of the core and rails stays with the open decisions in Section 12.

Execution roadmap

A standing constraint applies to everything below. Existing pipelines, workflows and partner deliveries are not disrupted, nothing destructive touches main branches, cardholder data is never in an agent's reach, and every step is incremental and reversible. The factory is built alongside current delivery, never as an all-or-nothing cutover.

The sequencing follows the whiteboard direction from the discovery and the climb order that has worked in our engagements: assessment first, a small pilot team of motivated volunteers, foundations before features, structured education over weeks rather than a one-day workshop, and tool and guardrail decisions before licences expand. Everything downstream depends on context, so context is built first, and it is built cheaply by extracting it from the one artefact that is always true, the code.

The moves, in order

The workstreams run in parallel, but the moves have an order. Steps 1 and 2 are prerequisites, and skipping them is how transformations end up justified by anecdote.

- 01 Make it an explicit leadership decision.** The three commitments in Section 12, made before more tooling gets built.
- 02 Establish a measurable baseline.** The pilot team's current flow, measured on the week-one instrument agreed in Section 13. This is what protects the programme from being judged on generated-code volume.
- 03 Run one change end to end.** One real bounded partner change through all the gates, verified by the new suite, with the certification pack produced by the pipeline.
- 04 Build the minimum factory around it.** Only the capabilities the pilot actually needs, delivered as a Golden Path. Never a slow central queue, and fast local feedback is preserved.
- 05 Take the pilot out of the release window.** A controlled path for low-risk, independently deployable changes, using risk classification, automated evidence, flags, canaries and rollback. The window stays temporarily for tightly coupled core flows only.
- 06 Begin structured core reduction.** Identify a high-value core behaviour, protect it with API-level or golden-master regression tests, define a stable seam, reimplement it as a service, migrate consumers gradually, and remove the legacy implementation when confidence is sufficient. Everything not yet reached stays stable and patched, because the business still runs on it. New core dependencies require explicit justification.

- 07 **Scale through cohorts.** Successful patterns become platform templates (Reference Architecture, TF-27). The next two or three teams get onboarded by the previous ones, in planned waves rather than all at once (Gradual Onboarding, TF-32). Practical education happens on real delivery work. Adoption and outcomes get measured by team.

The three phases

Phase 1. Factory MVP weeks 1-12 · builds Stages 3-5 · drafts the team structure

The first phase starts building the Software Factory itself, the shared platform every later capability runs on, and the first capability stood up on it is the specification and verification path, which runs from intake up to a release gate the pipeline can evidence. A small crew of motivated volunteers runs it, because the first job is to win hearts and minds with something simple but immediately valuable, not to convince the doubters yet. Generate living specifications from the existing codebase and the existing analytical documentation on every merge, in a format agents can consume, alongside a verification harness that extends the existing contract-test work and produces the certification pack. On top of the specs, build a prototype intake loop so intent is agreed with the partner before the factory generates production code. To prove the specifications are usable, one bounded change is taken through the existing SDLC and out of the release window. Deliverables are the first version of the knowledge service with two consumer groups live, a working docs pipeline, a first automated regression suite and evidence pack, a prototype intake loop, and a baseline of cost per change and percent customisation alongside the throughput metrics that every later phase is judged against. It follows Form the Core Team (TF-13), Thinnest Viable Platform (TF-19) and Proof of Concept (TF-6), runs inside the already-approved tool stack so it does not wait on new approvals, and builds on the spec-first pilot rather than starting fresh.

Phase 2. Lights-on factory ≥ 6 months · stands up Stage 6 · confirms the team structure

Build the secure pipeline described in Appendix A and scale agentic implementation and verification on top of the spec and test assets from Phase 1. In component terms: declarative versioned assembly lines, deterministic validation stages with Northvale's PCI and scheme checks wired in, the independent review agent, gated merging with logged deferrals, and the attribution and measurement layer, all hardened against Northvale's own compliance requirements and reviewed with its auditors. This is also where the work reaches the ledger core and component extraction, the oldest and most defended code, and where the most reserved practitioners get involved, because showing the factory works there is what convinces them. Comparable machine-readable specs across programmes expose real commonality, so reuse and rails-pack decisions get made on evidence rather than negotiation, discovered by diffing specs rather than argued in meetings, and shared rules, skills and standards become versioned factory assets with an owner. The pipeline is an Agentic Architecture (AN-4), and the rollout follows Gradual Onboarding (TF-32), the guard against Premature Scaling (AP-11). This is also where the stable team structure is confirmed in practice: the core and rails ownership drafted in Phase 1 is validated on real cross-programme work and adjusted before it hardens.

Phase 3. Paired onboarding [~6 months](#) · [tunes and transfers](#)

With the factory built and proven on one stream, most of the machinery already exists, so the remaining work is bringing new teams and components onto it rather than constructing anything new. We do this paired, roughly three to four onboardings together over about six months, at a pace that does not overload the team. The first real onboardings are where the rules, skills, specs and gates meet code they were not written for, so doing them jointly is how the system gets tuned against Northvale's actual repositories rather than a clean-room ideal. The engagement tapers by design: we lead the first, co-pilot the next, and Northvale runs the last while we stay on call. This is where the transformation of minds lands, skills transfer by doing real work together, not by documentation handoff. It follows Gradual Onboarding (TF-32) and leaves the capability owned by Northvale. Once several programmes run on the factory, its machine-readable specs let it surface commonality across them automatically, and the same factory can then extract those shared parts into the core and rails layers, strangler-pattern, on the second programme's budget, so cross-programme reuse becomes a by-product of onboarding rather than a separate rewrite programme.

The exit, and how to commit [commit to less or more](#) · [~1 year, 2-2.5 people](#)

The starting point never changes, so Northvale can commit to only the specs and verification, to specs extended into intent validation, or all the way through to agentic coding, committing to less or more without redoing the beginning. The full picture to weigh alongside the twelve-week first phase is roughly a year of work with two to two and a half people from re:cinq. And the shape of the whole engagement is an exit. By the final phase Northvale runs the factory and we step away. That is not a caveat, it is the point: we leave you owning what we built together (Finish the Transformation, TF-41).

Continuous. Adoption and measurement [from week 1](#)

A named factory owner, a lightweight internal practice to spread what works, and a monthly scorecard on the agreed metrics, including the eight-question diagnostic re-scored each quarter so progress up the ladder is visible on the same instrument that placed Northvale at the start. Each phase only starts when the previous one has a working, demonstrated result on real work, so the factory spreads from proof, not from pitch (Demo Your Ideas, TF-7). At each phase boundary, every remaining ceremony is reviewed against one question, does the factory now provide what this meeting exists to carry, and retired when the answer is yes, with the retro and the engineering guild as the standing exceptions.

How the engagement runs

re:cinq works in three phases, two of them fixed. Northvale has completed both. Discovery was the two days on site, the questionnaire and the delivery extract. The detailed plan is this report: it carries the diagnosis, the target model, the decisions, the twelve-week slice and the sequence, and it fixes the scope of delivery. Delivery is the twelve-week slice and what follows it, with team shape, duration and cost arriving as a separate proposal, scoped against this plan.

01

Embedded, not adjacent

We work inside the pilot team and inside Northvale's own stack. No parallel workstream, no artefacts that need handing over at the end.

02

Fixed scope before commitment

Delivery is quoted against a plan both sides have read, not against an estimate made during discovery. That plan is this document.

03

Training is part of the work

Education happens on real delivery work with the pilot team rather than as a course alongside it. A six-module programme, ending in train the trainers, is available where a cohort needs the ground covered faster than the pilot covers it.

{delivery plan}

each phase begins when the one before it proves out

kickoff

wk 12

Stage 6 ◆

steady state

Phase 1

Factory MVP • weeks 1–12

3–5

Phase 2

Lights-on factory • ≥ 6 months

6 | Lights-on factory

Phase 3

Paired onboarding • ~6 months

Tune & transfer ×3–4

Continuous

Adoption & measurement • always on

Owner, shared practice, monthly scorecard, quarterly re-score

Read as a delivery plan: phases run left to right on the timeline and each starts only when the previous one has a demonstrated result on real work. The bars darken up the adoption ladder, light foundations first, the lights-on factory, then paired onboarding that tapers as Northvale takes ownership.

Section 15 • People

Organisation and adoption

This section carries three messages. Adoption needs a named owner and a design. The roles after the factory have to be named before people are asked to build it. And the planning model has to give the factory room, or it becomes one more thing done in the gaps of a full workload.

Adoption has to be designed

Adoption is the organisation's own top risk, and it is currently the least designed part of the initiative. It needs a named owner with protected time rather than distributed goodwill. Distributed responsibility is exactly what produced the current islands. The pilot team works alongside the gateway team instead of receiving from it. Communities carry patterns outward, and the engineers proposed themselves that the people who built the islands be given a visible consultant role so their contribution is valued and the exchange is structured. The existing engineering guild becomes the venue, upgraded rather than replaced, with a format that lets the quiet majority speak. Hands-on boot camps in a closed

environment serve the middle group, the people who want to move and do not know how. And teams own their part of the factory rather than being handed it. Across interviews people described executing their programmes more than owning them, and a capability rolled out top-down will inherit that.

Roles, named early

The factory changes the chain. Analyst and compliance reviewer converge on a role that specifies changes and validates outcomes, with the tools doing what is done by hand today. The architect owns the layer boundaries and the contracts. Developers move toward maintaining the factory, the core seams and the hard cases, and toward the jurisdictions waiting in the pipeline. The certification tester role, as one lead put it, ideally disappears into pipeline-generated evidence, and the person freed writes certification criteria into specifications. The release manager becomes the person who designs flow and risk classes rather than runs the window.

None of this is settled and nobody can settle it in advance. What leadership can do now is commit publicly that the factory is not a headcount programme, that everyone on the pilot team has a place in the organisation after it, and that progression will be redefined so that seniority is not measured by hand-written throughput. Leadership also owes people a second answer: where the recovered capacity goes, and the answer is the jurisdiction pipeline and the backlog nobody has had time for, not out of the department. Saying that publicly is the concrete change-management work (Cultivate Psychological Safety, TF-15).

Planning has to give room

A team planned at one hundred percent cannot build a factory. The pilot team is planned at eighty percent for partner work and none at all for the first six weeks. Leadership decides what improvement capacity is for and stops using it as a buffer, which is the ring-fence decision in Section 12. Whether the current quarterly planning cadence survives the factory is an open question, and the pilot team is allowed to run on a shorter cadence to find out.

The experience gap

The gateway team achieved something real, with agent-generated contract tests in CI and measurable acceleration on the components it touches. And the way the wider set of islands is being built shows an organisation constructing its first system of this class without anyone who has built one before. That is the expected condition. Almost nobody has this experience. The signs are first-build patterns rather than talent problems: context infrastructure reinvented per team instead of shared, prompts promoted from a personal set-up with no versioning path, human involvement inherited from developer practice instead of designed as a control loop, and adoption left implicit while adoption is the organisation's own top risk. Every one of these is preventable with pattern knowledge rather than with more effort. Northvale has the talent, the openness and the momentum. What it does not have, and could not be expected to have, is experience with this class of system. Keep the islands, keep their inventions, and pair them with experience, so that the MVP is a second-system effort rather than a first.

The factory is a product

The factory needs a product definition, named users, an owner and a backlog, the same standard the organisation applies to anything it ships to a partner (Platform as a Product, TF-24). Its first users are the developers of the connector teams and the analysts already drafting specifications with AI. Its measure is whether the next team chooses it over building its own.

Measurement and governance

The ultimate outcome is annual business output per team capacity: delivered business value per year relative to the organisation's size. Cost per delivered change remains the factory's working unit metric. Around them sits a small set of operational measures, baselined in the MVP and reviewed on a monthly scorecard, with the eight-question ladder diagnostic re-scored each quarter on the same instrument that placed Northvale at the start (Measure What Matters, TF-36).

Group	Metrics
Flow	Intake-to-approved-specification time, intake-to-first-production time, merge-to-production time, share of changes shipped outside the release window, number of human handovers per change, time awaiting compliance and certification.
Quality	Rework caused by misunderstood requirements, automated verification coverage by risk class, certification evidence produced by pipeline versus by hand, escaped defects, change failure rate, corrective share of deployment requests, and recovery time. Recovery time already has a measured floor: P1 MTTR averaged about 70 minutes over the past eighteen months, stable, at roughly 9 to 21 incidents a month. That baseline is held, not improved. If it degrades as release frequency rises, the quality floor is being broken.
Economics	Cost per delivered change, certification effort per change, maintenance effort per programme, AI spend against delivered value, concurrent partners supported per team capacity.
Boundaries	Percentage of partner-specific versus reusable functionality per partner, reused versus net-new components per programme, rails packs shared across partners in a jurisdiction.
People	Partner experience, employee experience, adoption breadth across teams, escalation load concentration, onboarding time to a first safe production change.

PR counts, token usage and generated-code volume are not success metrics. Throughput metrics (lead time, change failure rate, time to restore) are tracked underneath all groups.

There is one economic caveat. Partner pricing today is estimated from development cost, not feature value. As the factory drives cost per change down, the pricing model will need to change with it. Otherwise efficiency gains translate into lower invoices rather than higher margin or capacity.

```
{the ambition, stated plainly}
```

Near term, first production in two and a half to three months instead of four to five, assuming fewer iterations against an executable spec, unchanged partner response times, and regression coverage and certification evidence built on the code actually being changed. The low-risk change class out of the release window by the end of the MVP. Longer term, one month to first production, as a conditional aspiration that depends on the layer boundaries holding and the factory maturing. All of it is measured on the same baseline established in MVP week one.

Risks, and the workstream that answers each

Taken from the organisation's own pre-mortem workshop.

Risk	Answer
Token consumption triggers a budget protection protocol	Token spend tracked against delivered changes from week one, a token-efficient harness as the default, and model access negotiated for the pilot rather than discovered as a blocker
The factory becomes a fifth island	The one-factory commitment, the authors of the existing islands on the build, and the next team's adoption as the exit criterion of Phase 2
Adoption of what is built not happening	A named adoption owner, team-to-team onboarding, and teams owning their part rather than receiving it
Added to existing workloads and starved	Named deferrals and a ring-fenced capacity share, decided before build
Audit and compliance exposure	Scoped agent identity, sandboxed execution, an immutable audit trail, cardholder data out of scope, and the evidence pack reviewed with the auditors in Phase 2
Security leaks	Scoped credentials, secret stripping before capture, egress restricted by policy
Building the wrong things faster	Specification gates plus partner validation before development
Loss of control over what is built	Humans at intent and release gates by design, and autonomy earned by evidence
Regression and quality	Verification automated before generation is scaled, with the MTTR floor held
Motivation and opposition	The career-ladder answer stated publicly, practice on people's own code, and recovered capacity going to the jurisdiction pipeline rather than out of the department
All eggs in one basket	A portable, open reference architecture and no single-vendor lock in the context layer

Section 17 · Beyond delivery

AI inside Northvale's payments products

The factory workstream is about how Northvale builds software. A second, separate question is what the software does: AI inside the products Northvale ships to merchants and partners. This assessment did not evaluate it in depth. It belongs in the strategy nonetheless, because merchants and partners everywhere are building AI into their operations and a payments platform whose customers use AI will be expected to support it. Both workstreams are necessary for future competitiveness, they should be planned separately and can proceed in parallel. Time should be set aside during the MVP for leadership, stakeholders and re:cing to discuss it.

The opportunity areas, in the order most payments platforms approach them:

- **Merchant onboarding and KYB.** Document extraction, classification and risk pre-screening across the paper-heavy flows every onboarding runs, with a human decision at the end.
- **Reconciliation exceptions.** Matching, explaining and proposing resolutions for the unmatched residue that today consumes analyst time.
- **Disputes and chargebacks.** Evidence assembly, categorisation and draft responses for human review, against scheme deadlines.
- **Fraud and risk assistance.** Surfacing precedent, explaining model decisions and flagging anomalies for human risk analysts, alongside the existing rule and model estate.
- **Merchant and partner assistance.** Conversational access to settlement status, payout timing, fee questions and integration guidance.
- **Regulatory and operational controls.** Monitoring, audit trails and model governance that make the above defensible.

Several of these fall into the EU AI Act's high-risk categories depending on their exact purpose and decision impact, notably where decisions touch creditworthiness, access to financial services or personal data, and all of them sit inside the operational-resilience obligations that already apply to a payments institution. That is a design constraint rather than a prohibition: there are established ways to build such systems compliantly, and expertise embedded in the platform can bring this functionality to merchants faster and more securely than each merchant building alone. We are in conversations with several financial-services organisations about AI both in software development and in their products, and can bring that perspective when Northvale wants to open this workstream.

What the factory is made of

The recommendations above describe what the factory must do. This section describes what it is made of, so that "software factory" does not stay an abstraction in a report. re:cinq builds and runs its own factory platform, Lore (github.com/re-cinq/lore, Apache 2.0), shared context infrastructure plus an agent pipeline, built around Claude Code as the agent runtime and running on Kubernetes. Lore serves as the reference implementation for this section, not a mandated product. Northvale's factory won't be Lore, it has to do the same jobs, and because the reference is open source, every component named below can be inspected, adopted, forked or rebuilt on Northvale's factory. Each component maps to a ladder stage, so the build order is the climb order, and none of it is speculative, all of it runs today. On tooling, the factory will use tools that already exist and are approved at Northvale unless they are not fit for purpose. Where a new tool must be introduced, the re:cinq project lead raises it with Northvale's security and compliance functions as early as possible so approvals never block the work.

`{one vocabulary}`

The Factory is the whole platform. A Floor is the long-running coordinator that dispatches work, one per team or trust tier. An assembly line is the fixed graph of steps a task passes through. A station is the isolated sandbox that runs exactly one agent, and an agent is a single ephemeral model run with a prompt and context. The factory contains floors, floors run assembly lines, assembly lines are made of stations, stations run agents. The hierarchy matters because governance attaches to it: permissions, identity and audit live on stations and floors, never on a developer's laptop.

```
{factory architecture}
```

Developer machine

Agent client

The developer's coding agent, reaching shared context and the pipeline through an MCP server.

Factory MCP server

Context retrieval (context files, ADRs), hybrid search, persistent agent memory, task dispatch, memory proxy so local learnings become organisational knowledge.

Kubernetes cluster

Context service

Context, memory, pipeline and ingest APIs.

Factory agent

Task worker, sandboxed agent jobs, multi-agent lines, scheduler for reindex, gap detection, spec drift and evals.

Web UI

Repo dashboard, onboarding, pipeline view, cost tracking.

PostgreSQL + vector store

Shared context, agent memory, pipeline tasks and jobs, repo registry. Teams isolated at schema level.

Source-control app

Read repos, open pull requests, set secrets, under a scoped identity per client.

Reference factory platform. The agent on the developer machine reaches shared context and the agent pipeline through an MCP server, and the cluster runs the context store, the agent workers, the scheduled context jobs and the source-control integration that every team consumes.

3 Standards and review that scale

Repo onboarding is itself an agent task. Point the factory at a repository and it inspects what exists, generates only what is missing, a machine-readable context file with the repo's conventions, architecture decision records inferred from the code, a structured pull request template, and the CI checks that enforce it, then opens a PR for the team to review and merge. This is how "standards enforced in tooling" reaches 140 repositories in weeks instead of quarters, and it is safe to run against repos that already have some of these files. The second Stage 3 component is automated first-pass review. Every agent PR is checked against the spec and the conventions before a human sees it, with a bounded fix loop, up to two automated iterations on requested changes before the task escalates to a person. The bound is deliberate, it turns review from the bottleneck into a gate.

4 The context layer and the security floor

One shared context store for the whole organisation. Repository content, context files, ADRs, runbooks, specs and code, is chunked, embedded and indexed on every merge to main, with a nightly full reconciliation so nothing drifts or orphans. Retrieval combines semantic and keyword search, and a knowledge graph tracks which service uses what and who owns which part. On top of the store sits organisational memory. Decisions and lessons are written back at session end, facts are extracted automatically, contradictions invalidate old facts, every fact carries a confidence tier, and secrets are stripped before anything persists. Two scheduled jobs keep the context honest: gap detection finds undocumented areas and files gap-fill tasks, and spec drift compares the specs against the actual code every week, which is the "living specifications" recommendation as a running job rather than a one-off project. Everything is assembled on demand into a single token-budgeted briefing, so every

developer and every background agent starts a session with the same organisational knowledge. This is what converts the knowledge-recovery tax, today paid by interrupting a senior engineer, into an API call.

The security floor is built at this stage too, and in a PCI environment it is the part the auditors will read first. Agents run in ephemeral isolated pods, non-root, with egress restricted by network policy. No credentials are stored anywhere, identity comes from the platform, tokens are scoped per client and rate-limited, and the store isolates teams at the database schema level, the same mechanism that keeps Northvale's tenants isolated from each other. Cardholder-data repositories and environments are classified and held at the lowest trust level, and no agent station has a route into the cardholder-data environment.

5 Intake where people already work

The factory accepts work from the surfaces people already use. A label on a ticket dispatches a task. A slash command in the team's chat channel creates one, and the factory posts back to the same channel when the PR is open or a task fails. A plain-language form lets an analyst describe a partner change with no technical format at all. The factory fetches the repo's context and generates the spec, the data model, the certification criteria and the task breakdown in that repo's own conventions, then opens a spec PR for an engineer to review. The engineer reviews intent, not prose, and once the spec merges, implementation runs against it task by task. This is the "partner validation before development" recommendation in running code, and the same generated spec is the artefact the partner signs off before development starts. Task state stays visible in normal channels, every task opens a ticket on the target repo and reports its outcome there, so nobody needs a new dashboard to know what the factory is doing. Scope expands on evidence. Each repo carries a trust level, documentation first, then tests, then implementation, then full, promoted automatically after a run of successful merges and held low deliberately for regulated repositories.

6 The assembly line itself

Assembly lines are declarative, versioned definitions, a fixed graph of four node types, generate, validate, gate and retrospective, stored as configuration that is reviewed and diffed like any other code. This is the "pipeline as versioned configuration" recommendation made literal, one concrete artefact for security, compliance and scheme reviews to approve. Validation is deterministic and runs before any human looks. Lint, type checks and tests are mandatory stages with one automatic fix retry before escalation, and this is where Northvale's security scanning, PCI checks and scheme certification criteria slot in as further fixed stages, producing the evidence pack as a by-product. Review is done by a second agent with fresh context and no knowledge of the generation session. Merging is gated. Low-blast-radius changes, documentation, specs, context files, can auto-merge behind green CI plus an approval plus a trust threshold, and every deferral is logged with the full rule trace, while code changes always keep a human at the merge gate, which is the Stage 6 rather than Stage 7 design expressed as mechanism. Every action is attributed to the human who delegated it and the agent that executed it, in an immutable, redacted audit log. Changing the dangerous settings, enabling autonomy or widening the auto-merge paths, requires two keys, an administrator and a code owner of the affected repository approving through a labelled PR. Measurement is built into the same platform: cost and tokens per task, throughput and success rates per repo, retrieval latency, and outcome feedback, where context that contributed to merged PRs is strengthened and context behind rejected ones is weakened, so the factory's knowledge improves from its own results. The monthly scorecard in the roadmap reads from here, not from a spreadsheet.

Adopt, fork or rebuild

Because the reference implementation is open, the choice is Northvale's per component. What must be Northvale-specific in any case: the PCI and scheme gates and scanners, the classification of regulated repositories and environments, the mapping of isolation boundaries onto tenants, and the integration with the existing CI and environment layer, the one part of Northvale that already runs at Stage 6 discipline. What should not be rebuilt from scratch is the pattern set above, since it encodes the expensive lessons: bounded fix loops, trust tiers, deferral logging, two-key changes to privileged settings. One detail is worth naming because it mirrors this report's own conclusion. The reference platform ships a full-autonomy mode and keeps it off by default behind two independent switches, one per repository and one per cluster. The capability existing and the capability being on are separate decisions, and Stage 6 with humans at the gates is the target here.

Pattern references

Nothing in this report's recommendations is novel. Each one corresponds to a named pattern in the re:cinq transformation pattern library, part of Waves of Innovation and published at re-cinq.com/pattern-library. The cards below are the ones this assessment leans on most, with the section that applies each.

Transformation patterns

TF-6

Proof of Concept

Section 13. The MVP proves the specification and verification path on one team before anything scales.

TF-7

Demo Your Ideas

Section 14. Each phase starts from a demonstrated result on real work, not from a pitch.

TF-11

Executive Commitment

Section 12, the sponsor. The capacity decision cannot be delegated.

TF-12

Appoint Transformation Champion

Sections 12 and 15. Named adoption ownership instead of distributed goodwill.

TF-15

Cultivate Psychological Safety

Section 15. Roles and headcount stated plainly rather than managed around.

TF-13

Form the Core Team

Sections 12 and 13. The pilot crew as a dedicated cross-functional group.

TF-19

Thinnest Viable Platform

Section 13. Only the capabilities the pilot change needs.

TF-20

Delayed Automation

Section 10. Automate checks once they are stable, not while they are in flux.

TF-23

Staged Funding

Sections 12 and 13. Funding released against milestones, and the week-twelve scale, adjust or stop decision.

TF-24

Platform as a Product

Sections 10 and 15. Teams join the factory when it beats their current way of working.

TF-27

Reference Architecture

Section 14 and Appendix A. The open reference platform, adopted per component.

TF-31

Establish Team Topologies

Sections 1 and 7. Shaping teams so the layered architecture is the path of least resistance.

TF-32

Gradual Onboarding

Section 14. Cohorts onboarded by the previous team, in planned waves.

TF-36

Measure What Matters

Section 16. Throughput and outcome metrics, never activity metrics.

TF-45

Minimize Cognitive Load

Section 10, the Golden Path, and the engineers' own asks in Section 15.

TF-47

Enable Self-Service

Section 10, provisioning inside the pipeline instead of tickets.

TF-41

Finish the Transformation

Section 14. The exit and the defined reduction schedule.

AI native patterns

AN-9

Retrieval-Augmented Generation

Section 10, the shared context before an agent's first step.

AN-4

Agentic Architecture

Section 14, Phase 2. The pipeline as floors, assembly lines and stations.

AN-10

Intent-Driven Development

Section 10. Partner validation against a generated spec and prototype before development.

AN-14

Human-in-the-Loop Validation

Section 10. The permanent decisions at intent and release.

Pitfalls and anti-patterns this report is written to avoid

AP-4

Incremental Facade

Diagnosis pattern 6. Why the factory cannot be built in the gaps of a full workload.

AP-11

Premature Scaling

Section 13. Why the first move is one team rather than a rollout.

AP-32

Accidental Transformation

Section 7. Why the capacity and ownership decisions come before more tooling.

AP-54

Human-Last Collaboration

Section 10. Why humans sit at the gates by design rather than being consulted after the fact.

Appendix C

External evidence

Four external sources bear directly on the findings. They are included because each one predicts something this assessment observed independently at Northvale.

DORA 2025, State of AI-assisted Software Development

Nearly 5,000 technology professionals. AI works as an amplifier, magnifying the strengths of strong organisations and the problems of struggling ones, and AI adoption alone does not improve delivery throughput or stability without the surrounding capabilities. This is the evidential basis for sequencing verification and specifications before broad autonomy.

Faros AI, The AI Productivity Paradox

Telemetry from 10,000 developers across 1,255 teams. Heavy AI use produced 21% more tasks finished and 98% more pull requests merged, while lead time, failure rate and recovery time stayed flat. Review time rose 91%, pull request size rose 154%.

METR, 2025

Randomised controlled study with experienced developers on familiar code. With AI assistance they were around 19% slower while estimating themselves about 20% faster. Perceived and actual productivity can diverge widely, which is why the factory is measured on throughput, not sentiment.

Melvin E. Conway, How Do Committees Invent? (1968), and the Inverse Conway Manoeuvre

System structures tend to mirror the communication structures of the organisations that build them. The basis for shaping teams around the target architecture rather than letting an inherited org chart dictate it. Cited here as TF-31, Establish Team Topologies.

re:cing, The AI Adoption Ladder (2026)

The published framework this assessment applies, including the field distribution across stages and the eight-question diagnostic used above.

Appendix D

Glossary

Each term is defined once, here. The delivery-measurement definitions are Northvale's own, taken from the data extract documentation, so the numbers in this report mean what Northvale means by them.

Term	Definition
Initiative	The unit of work used throughout this report. Larger business and technology initiatives moving through the delivery lifecycle, rather than individual tickets.
Delivered initiative	An initiative whose tracker status is Done and which has a confirmed delivery date. In this extract the confirmed date matches the actual transition to Done for all 412 initiatives.
Lead time	How long an initiative takes. Three spans are used. Discovery to implementation runs from the moment an initiative first enters Discovery to the moment implementation first starts. Implementation to done runs from that start to the confirmed delivery. Total lead time is the full span.
Complexity rating	Northvale's T-shirt sizing of an initiative's inherent complexity, XS to XL. It reflects capabilities to build, regulation, schemes, business rules, third-party dependencies and cross-team coordination. It explicitly excludes difficulty caused by the current technical landscape or organisation.
Ledger core	The original processing monolith: ledger, settlement engine, message handling and batch processing. Stable, widely reused, unevenly covered by automated tests, and the reason the release window exists.
Gateway services	The modern, independently deployable services estate around the core, roughly 140 repositories, deploying on demand.
Connector	The integration between the platform and one acquirer, bank or partner system. Thirty-one are live. On paper they share an interface; in practice 40% to 70% of the code is adapted per partner (interview estimate).
Rails pack	The local payment methods, scheme variations and regulatory reporting for one jurisdiction. Today forked per partner; in the target model a versioned asset per jurisdiction.
Release window	The four-weekly batched release of the ledger core, with a one-week certification freeze, roughly a week of manual regression and staggered deployment into about 30 partner and jurisdiction environments.
Corrective deployment	A patch to acceptance or a fix to production, measured as a share of all deployment requests in a window.

Term	Definition
Certification evidence	The test results, traces and screenshots a card scheme or internal compliance requires before a change touching scheme messaging or settlement can go live. Today assembled by hand; in the target model produced by the pipeline.
Software Factory	Northvale's name for the system this report recommends building. People, process, platform and governance covering the path from validated intent to measured outcome. Not agents, and not code generation.
Strangler pattern	Route consumers to a new implementation behind a stable seam, then retire the old one. Not a golden hammer. Where two domains are coupled because boundaries were never challenged, only setting correct boundaries fixes it, and careless strangling produces a distributed monolith.
Human gate	A point where a person decides, reviewing durable artefacts rather than watching work happen. This report places two permanent ones, at intent and at the release decision.

We leave you owning what we built together.

This assessment is the first step of an engagement designed to hand the keys back. Every asset the factory produces (specs, pipelines, rules, tests, evidence, measurement) is built inside the client's own stack, owned by the client from day one.

re-cinq.com

Pini Reznik, CEO, pini@re-cinq.com

Michael Müller, CTO, michael@re-cinq.com

