

LAPS Enterprise Framework

Predictable software delivery with human and AI teams

Version 0.8 | Public beta | 14 October 2026 | Enterprise framework and lighter team route

Learn, Act, Prove, Sharpen

By Gaurav Rajwanshi

LAPS stands for Learn, Act, Prove, Sharpen. These four actions define the recurring operating cycle of the LAPS Enterprise Framework. A lap is a focused, bounded cycle that produces evidence and an accountable next decision. One day, and half a day where readiness and review capacity allow, are operating options. Release approval and observation of business outcomes retain their own timing.

This public beta is available for practical use, adaptation, and review with attribution to Gaurav Rajwanshi and the version used. Third-party source material retains its own terms. The paper is a proposed operating framework, not a certification or an independently validated performance study. Its recommendations should be piloted and adapted to the consequences of each use case.

This edition draws on the author's work with high-profile enterprise clients over the last two to three years, published professional reflections, private practitioner discussions, and the public sources in Section 23. Client identities and confidential delivery details are intentionally excluded. Examples are illustrative, and no client endorsement is implied.

The canonical edition and review channel are at <https://www.gauravrajwanshi.com/laps/>. Please identify version 0.8 when sharing feedback. Future revisions will have their own dated change record.

This framework explains how leadership and delivery teams can use AI to shorten the path from intent to verified feedback while improving predictability and controlling consequential uncertainty and unwanted variation. It combines a manifesto, a delivery operating model, role mappings, daily and half-day operating cycles, practical examples, and reusable templates.

Contents

- 1 Leadership guide
- 2 Manifesto for LAPS
- 3 The LAPS cycle and the uncertainty loop
- 4 Three AI overlays
- 5 Proportional control
- 6 Forecasts and commitments
- 7 Roles and decision rights
- 8 How teams work together
- 9 The team digital brain
- 10 The end to end lifecycle
- 11 Engineering with digital workers
- 12 Engineering AI behaviour and data
- 13 Testing and evaluation
- 14 Users interacting through AI
- 15 Release control and operational response
- 16 Cadence and measures
- 17 Banking and insurance examples
- 18 Enterprise and lighter team versions
- 19 Adoption and governance
- 20 Reusable working templates
- 21 Sources of the operating model
- 22 Glossary
- 23 Sources and interpretation

1 Leadership guide

Recommendation. Adopt LAPS as the operating framework for software built with AI, software containing AI, and software used through AI. Use short learning and execution loops inside a stable agreement about outcomes, decision rights, delivery forecasts, quality, cost, and acceptable exposure. The objective is faster verified learning and dependable delivery and operation, with fewer avoidable surprises. Shorten the interval between intent, evidence, and the next useful decision. One-day cycles, and half-day cycles where the work is ready, are explicit operating options.

The central management question is: **What do we know now, what remains uncertain, and what evidence justifies the next commitment?** Faster artifact creation is valuable only when it improves the complete path to an accepted, usable outcome.

What leadership should be able to see

For each initiative, provide an agreed outcome and minimum useful scope; the current delivery forecast and its basis; the next evidence milestone; the few unknowns most likely to change the decision; dependencies and accountable owners; cost and exposure limits; and the decision needed from leadership, with its deadline. Report actual benefit separately from readiness to release. A functioning release can still fail to improve the business.

Approve a bounded pilot of the method before setting organization-wide productivity or staffing targets. Require evidence of forecast reliability, quality, total cost, review capacity, customer impact, and recovery. A shorter coding cycle alone does not justify fewer specialists or a more aggressive delivery promise.

What changes from earlier delivery approaches

Phased delivery creates explicit plans and review points. It can suit stable requirements or irreversible dependencies; a completed phase still cannot remove an untested assumption. Agile approaches bring customer feedback and adaptation closer to the work. The Agile Manifesto and its principles emphasize collaboration, useful software, and learning; Scrum explicitly uses empirical inspection and adaptation to improve predictability. [16][17] Leadership uncertainty often arises when teams have no shared definition of a forecast or commitment, rather than from iteration itself.

AI can shorten drafting, building, and some evaluation cycles. It can also introduce variable outputs, hidden assumptions, supplier changes, and more work awaiting review. This framework retains planning and independent evidence while accelerating the activities that can safely move faster. Discovery may take days or weeks even when an agent completes a code change in minutes.

Use stages to organize decisions, small increments to learn, and automation to execute bounded work. Some approvals remain serial because their dependencies are real. Avoid interpreting the ten stages as ten departments or a requirement to finish the entire product's design before building any useful slice.

Reading routes

Reader	Start here	Decisions this supports
Executives and business decision makers	Sections 1 to 3, 6, 16 to 19	What to fund, what to expect, what to change, and when to stop.
Program and project managers	Sections 6 to 10, 16, 20	Forecasting, dependencies, capacity, commitments, and escalation.
Product, engineering, research, design, data, and assurance practitioners	Sections 2 to 15 and 20	Daily work, role collaboration, evidence, controls, and release readiness.

The main framework applies across banks and insurers; local obligations and delegated authority remain part of the organization's control system. The lighter route in Section 18 reduces administration for limited-impact work. Small team size does not lower the consequences of a critical use case.

2 Manifesto for LAPS

We build and operate software with people and AI to produce useful outcomes that others can depend on. We increase confidence by making uncertainty visible, testing important assumptions early, and keeping variation within agreed limits. We make commitments that evidence can support and explain promptly when that evidence changes.

In this manifesto, **assured** means confidence supported by evidence and accountable decisions within a defined scope. It does not mean guaranteed dates, identical AI outputs, zero risk, or a certification.

Six values

1. **Verified outcomes over output volume.** Progress means that a useful result works within agreed limits.
2. **Confidence earned through evidence over asserted certainty.** We expose what is unknown and make commitments proportionate to what we know.
3. **Small reversible steps over large untested commitments.** We learn before increasing cost, exposure, or autonomy.
4. **Shared intent and multidisciplinary work over repeated handoffs.** We bring the necessary skills to the outcome and preserve their judgment.
5. **Accountable autonomy over unchecked delegation.** People own decisions; AI works within clear, enforceable mandates.
6. **Continuous learning and renewal over approval as a finishing line.** We keep checking behaviour, value, assumptions, and recovery after release.

Plans, documents, specialist roles, and automation remain useful. These values guide decisions when delivery pressures compete. They do not waive legal obligations, agreed quality, or required independent approval.

Twelve principles

1. **Start with the outcome.** Define whose task improves, the current baseline, the intended benefit, and unacceptable harm.

2. **Make uncertainty visible.** Name consequential unknowns, assumptions, dependencies, and the decisions they affect.
3. **Shorten the learning cycle.** Use the smallest credible experiment or useful change that can inform the next decision. Move toward daily or half-day feedback where evidence and review capacity support it.
4. **Forecast honestly.** Publish scope, a delivery range, its evidence basis, conditions, and the next update; distinguish targets from commitments.
5. **Control the whole flow.** Limit started work, protect review and integration capacity, and resolve the oldest consequential blockers.
6. **Stabilize what must be dependable.** Keep permissions, business rules, data contracts, critical calculations, and acceptance standards explicit and testable.
7. **Bound what remains variable.** Measure variation, exclude unsupported cases, set limits, and provide abstention, escalation, and recovery.
8. **Combine skills without erasing accountability.** Reduce avoidable handoffs while retaining specialist judgment and required independent challenge.
9. **Give people and AI usable context.** Maintain approved, permissioned, current intent and evidence in shared knowledge.
10. **Earn and constrain authority.** Expand mandates only after evidence and approval; verify actions and resulting state outside the model.
11. **Test the complete service.** Evaluate software, models, integrations, people, user delegation, and recovery under realistic conditions.
12. **Improve the method with evidence.** Compare forecasts with actual outcomes, learn from failures and near misses, and change or retire what no longer works.

This is a proposed manifesto for this framework, inspired by the purpose of the Agile Manifesto and informed by the call and LinkedIn posts. It is not an established industry standard or a replacement definition of Scrum. [1][7][9][12][13][15][16]

3 The LAPS cycle and the uncertainty loop

The core is **Learn** → **Act** → **Prove** → **Sharpen**, repeated around a useful outcome. The name describes the work of each lap. A delivery integrator coordinates the cycle; relevant owners retain their decisions.

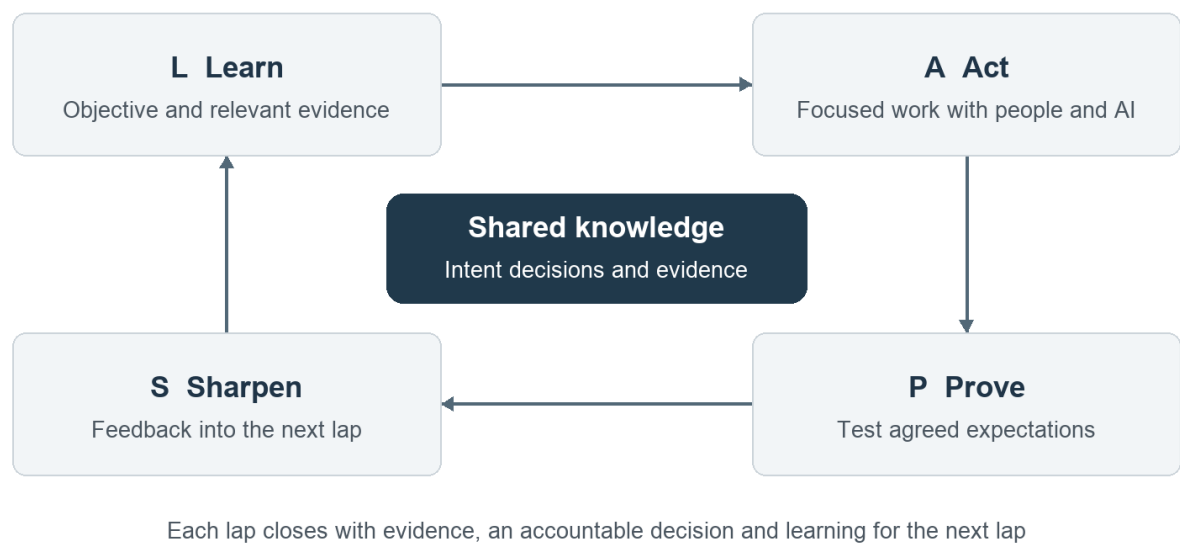
- **L — Learn:** understand the objective and relevant evidence.
- **A — Act:** carry out focused work with people and AI.
- **P — Prove:** test the result against agreed expectations.
- **S — Sharpen:** use feedback to improve the solution and the next lap.

In **Learn**, agree intent and limits, inspect relevant evidence, identify consequential uncertainty, and set expectations before acting. In **Act**, perform bounded work with people and AI. In **Prove**, examine independent evidence against those expectations, including failure cases and remaining uncertainty. In **Sharpen**, use feedback to improve the solution and method, record the accountable decision, update shared knowledge, and select the next lap.

Prove is scoped to the agreed expectations and available evidence. It does not mean mathematical proof, universal correctness, or zero risk. A failed test, a rejected hypothesis, or a blocked dependency remains a valid finding. Sharpen can lead to acceptance, revision, a stop, or a recorded

blocker; it does not imply automatic release. Checks and feedback can occur throughout a lap, and the four actions do not create four sequential departments.

The team digital brain supports the loop by connecting approved intent, assumptions, decisions, implementation, evaluations, forecasts, and operational results. It is the shared memory of the service. It cannot replace judgment or make incomplete evidence true.



Separate uncertainty from variability

Uncertainty is a gap in what we know. Examples include an unclear user need, an untested integration, or an unknown approval lead time. Research, experiments, and observation can reduce it. **Variability** is the spread of results or times: review duration, incoming cases, model responses, or cost per task. Some variation is avoidable; some must be measured and accommodated. **Predictability** means a forecast or behaviour expectation remains useful within stated limits. A consistent wrong answer is predictable but unacceptable.

Do not seek uniformity in every activity. Diverse ideas, legitimate customer needs, and independent challenge are valuable. Reduce unwanted variation in the path to an accepted outcome; preserve variation that helps discovery or appropriate service. Eliminate unnecessary uncertainty where feasible, bound what remains, and maintain a response to surprises.

Area	Example uncertainty or variation	How to reduce or contain it
Value and intent	Different interpretations of the user problem	Observe real work, compare alternatives, agree outcome and examples.
Solution and data	Unknown integration behaviour or weak data coverage	Test contracts, inspect data, run a narrow prototype, record exclusions.
Delivery and coordination	Variable review time, unavailable dependency, scope growth	Limit active work, reserve capacity, agree dependencies, update forecasts.
AI behaviour	Different outputs, unsupported claims, variable tool choices	Version inputs and policies, evaluate repeated trials, enforce action limits.
Operation and change	Demand bursts, outages, supplier updates, staff absence	Monitor, rehearse failover, provide cover, control changes and recovery.

Keep an uncertainty register that leads to action

For each consequential unknown record the decision at risk, current evidence, assumption, owner, next experiment or dependency resolution, time and cost limit, and a review date. Use four plain states: **unknown, under investigation, bounded by evidence, or changed and reopened**. Describe the remaining exposure in words and relevant measures. Do not average unrelated unknowns into a single confidence score.

Prioritize an unknown when resolving it could change funding, design, release, or the forecast. A small test against the real dependency can be worth more than building more features. Stop investigating when the next decision has sufficient evidence, or when the cost of further learning exceeds its value and a permitted, reversible alternative exists. Critical unanswered questions can block release; they do not disappear because the timebox expires.

Record new unknowns as work proceeds. A rise in identified unknowns can reflect better discovery. Confidence may legitimately fall after a failed experiment. Never force a steadily improving dashboard, penalize disclosure, or claim that uncertainty always declines over time.

Set the evidence threshold for the next decision before conducting the test. The owner should be able to say what result supports continuation, what requires a different approach, and what blocks further exposure. High-impact uncertainty requires stronger evidence or a narrower mandate. Where probabilities cannot be estimated credibly, describe the scenario and consequences instead of manufacturing a number.

Operate at different speeds

Loop	Typical rhythm to tailor	Stable boundary
Agent execution	Minutes to hours, with automated checks	Task objective, permissions, budget, stop rules, evidence required.
Team learning and integration	One day; half a day for ready, bounded work; longer where evidence requires it	Outcome slice, acceptance contract, active-work limits, independent checks.
Delivery commitment	Weekly and immediately after material changes	Leadership agreement, forecast history, dependency commitments, decisions.
Service and portfolio oversight	Continuous incident response, periodic service and funding reviews	Exposure, resilience, customer outcomes, risk appetite, continued value.

Section 16 describes the daily and half-day operating cycle. A cycle ends with an evidenced decision; a release has its own authorization, and business impact may require a longer observation period. These are starting rhythms, not service promises. Reuse evidence between loops so leaders receive decisions and consequences rather than every prompt or tool call. Continuous assurance means producing evidence as work happens; it does not mean asking a committee to approve every small action.

4 Three AI overlays

Record three independent yes or no flags in the initiative passport. Multiple flags may apply. AI use during development does not automatically make the released application an AI product.

Overlay	Meaning and examples	Additional evidence required
D — AI in delivery	AI researches, writes stories, designs, codes, tests, reviews, or operates delivery tooling.	Approved tool use; scoped access; reviewed artifacts; provenance of important changes; independently executed checks.
P — AI in the product	A model predicts, ranks, generates, recommends, or plans as part of product behaviour.	Data and model records; behavioural specification; evaluation results; runtime limits; drift and supplier change monitoring.
I — AI in interaction	Users interact through an embedded assistant, voice agent, browser agent, or external personal AI.	Identity and delegation contract; scoped permissions; interaction tests; confirmation rules; receipts and human recovery.

Examples: a payroll website coded with AI can be D only. A forecasting service developed manually can be P only. A conventional booking API used by a customer's assistant can be I without P. An AI-built customer service agent with a conversational interface can require all three.

Classify at the boundary where effects occur. A delivery agent allowed to deploy production changes needs controls for that authority even if the application has no AI at runtime. An external assistant does not gain authority because it speaks fluently or claims to represent a user.

For each initiative, also record the users affected, jurisdictions, sensitive data categories, system dependencies, reversibility of actions, business criticality, intended lifespan, and accountable service owner.

Select the capability the outcome needs

The overlays identify where AI participates. Separately identify what capability the task requires. Following the business-first approach in the capability post, compare the smallest suitable solution with the current process and a conventional alternative. [11] The evidence choices below are this framework's implementation guidance.

Business need	Candidate capability	Evidence to prioritize
Estimate future demand or risk	Prediction or forecasting	Errors over relevant time periods, calibration where applicable, and the resulting business decision.
Route work or flag unusual cases	Classification or anomaly detection	Missed cases, false alarms, cost of each error, and review-queue capacity.
Answer from approved knowledge	Retrieval with answer generation	Source permissions, retrieval coverage, factual support, freshness, and abstention.
Draft or transform content	Generation or extraction	Accuracy, completeness, suitability, and human correction effort.
Complete a bounded business task	Deterministic workflow, or an agent where justified	Correct final state, permissions, retries, exceptions, and recovery.

Product integration describes how a capability is delivered; multi-agent orchestration describes how work is coordinated. Neither is automatically required. Add coordination only when measured quality or throughput justifies its additional cost and failure modes.

5 Proportional control

Impact classes

These are internal delivery classes, not legal classifications. Assign the highest credible impact and document the reasons. High uncertainty or highly sensitive data increases the evidence required. A low score on one factor cannot cancel a severe consequence on another.

Class	Typical consequence	Minimum review route
R1 — Limited	Reversible internal convenience task; little sensitive data; bounded disruption.	Named owner, peer review, automated checks, recovery method.
R2 — Material	Customer-facing work, sensitive records, operational decisions, or meaningful financial exposure.	Product, engineering, and assurance review; representative evaluations; staged release and monitoring.
R3 — Critical	Potential effects on rights, safety, essential services, major financial loss, or systemic operations.	Specialist and independent assurance, explicit risk acceptance, tested continuity, and required sector or organizational approvals.

Unlawful or prohibited use does not receive an internal exception. An owner records a stop decision and the reason. Risk classification is reconsidered when purpose, audience, data, tools, geography, or autonomy changes.

Authority levels

Record authority separately for delivery workers, product agents, and user agents. A high quality model does not justify broad permissions.

Level	Permitted behaviour	Boundary
A0 — Observe	Read approved information and report findings.	No changes or external commitments.
A1 — Propose	Prepare drafts, plans, or proposed changes.	A human or approved workflow executes them.
A2 — Act after approval	Execute the particular action a person approved.	Approval binds to the target, parameters, and expiry.
A3 — Act within a mandate	Execute defined classes of action within pre-approved limits.	Scoped identity, limits, monitoring, revocation, and escalation.

Avoid granting unrestricted autonomy. Approval for one action is not a continuing mandate. An A3 grant specifies resource scope, maximum value or volume, permitted recipients, time limit, and stop conditions.

6 Forecasts and commitments

Leadership needs reliable decision information even when the solution is still emerging. Establish a **delivery agreement** between the product owner, delivery team, service owner, and sponsor. It can be a view of the existing passport and release record rather than another document.

Distinguish the promises being made

Term	Meaning	What must accompany it
Target	A desired date or result	Business reason, flexibility, and consequence of missing it.
Evidence milestone	A timeboxed investigation or decision checkpoint	Question, method, resource limit, and decision date; a negative result is allowed.
Forecast	An estimate of delivery timing or scope	Defined scope, data or judgment basis, assumptions, range, and update date.
Commitment	An authorized undertaking for a defined outcome and scope	Evidence supporting the undertaking, capacity, dependencies, quality, fallback, and escalation terms.
Release authorization	Permission to expose an identified version	Accepted evidence, audience, authority, operating limits, and readiness.
Outcome achieved	Observed business benefit	Baseline, measured result, affected population, and observation window.

A forecast is not permission to release. A commitment requires active management but cannot make an uncertain event guaranteed. Use the precise state in reports instead of a percentage complete that mixes discovery, coding, review, and benefit.

Use three planning horizons

Explore. Fund a bounded learning question and commit to the investigation effort and decision checkpoint. Report scenarios rather than a precise release probability when evidence is weak. Do not promise that an experiment will prove feasibility.

Prepare. Once the minimum useful scope, important dependencies, acceptance method, and review capacity are understood, publish a delivery range. State which uncertainty still affects the range. Establish interfaces, reserve specialist capacity, and make scope choices before increasing concurrency.

Deliver. Make a release commitment only for a sufficiently understood slice within authorized risk and capacity. Keep the forecast current, protect essential scope, and record changes. Continue research on later slices without repeatedly changing the near-term slice. A discovered safety issue can still require a stop.

Build a forecast from the actual system of work

The flow and forecast lead works with the team and data analyst. Record when comparable work is ready, started, blocked, accepted, and released. Include review, integration, approval, release windows, and rework. Kanban's flow measures and service level expectation provide a useful basis for observing work and expected completion times. [18] The leadership agreement and scenario method below are proposed practices in this framework.

Use empirical distributions from comparable completed work where they are available. For a portfolio of remaining items, a suitable simulation can sample observed throughput and model required dependencies and calendar constraints. State the data window, item definitions, planned capacity, included waits, and assumptions. Test the method on past forecasts before relying on precise probabilities. Do not divide total points by a best-case AI speed multiplier, or add individual task percentiles and label the sum a program percentile.

Separate novel work, routine changes, and external approvals when their behaviour differs. Account for unfinished, aging work and interruptions; a dataset containing only easy completions can understate delivery time. When roles, tooling, scope mix, or review routes change substantially, reassess comparability. Use early AI-delivery observations to update the forecast, not to erase the pre-change baseline.

When history is sparse, use a reasoned scenario range with explicit assumptions and call it **judgment based**. Do not attach a P80 label merely because the team feels fairly confident. Increase forecasting precision only as evidence supports it. A team may know when it will resolve a dependency while still being unable to forecast release reliably.

Explain probability in plain language

If a fitted forecast produces a P80 date, it estimates an 80 percent chance of completing the stated scope on or before that date, conditional on its assumptions. P50 represents the median date. Neither says the software is 80 or 50 percent complete, and neither measures the probability that a customer outcome is correct. The interval between P50 and P80 is not an 80 percent confidence interval.

Choose the probability level according to the decision and consequences; there is no universal P80 release rule. For external obligations, assess the consequences of delay and a feasible contingency. Probabilistic schedule confidence never overrides a failed quality or authority constraint.

Calibrate confidence and preserve the record

Retain dated forecast snapshots, scope baselines, assumptions, and actual completion. Across a sufficiently large, comparable set, approximately eight out of ten P80 forecasts should complete by their stated dates. Treat small samples as uncertain. Review coverage alongside the width of ranges and the cost of buffers; extremely wide ranges can cover outcomes while being useless for decisions.

Define the calibration unit and forecast horizon in advance, such as each release forecast made four weeks before its target. Multiple weekly updates to one release are related observations; do not count them as independent projects to inflate the sample. Keep separate views for first commitment and later updates, and report changes in the forecasting method.

Report delivery against the original commitment and the latest agreed forecast. Show scope additions, removals, and changes explicitly. Explain forecast movement in terms of new evidence, capacity, dependencies, or scope. Silent rebaselining prevents learning and gives leadership false confidence.

Protect fixed dates through explicit choices

If a date is immovable, define the minimum acceptable outcome and mandatory controls first. Work backwards through real dependencies and release windows. Establish the latest dates for scope reduction, contingency activation, and escalation. Prioritize learning on anything that could invalidate that path.

If the forecast no longer supports the date, the sponsor chooses among a smaller acceptable scope, a proven simpler solution, added capacity at the actual bottleneck, an approved manual contingency, or renegotiation where permitted. Additional agents do not resolve missing data rights or absent reviewers. Never meet a date by concealing a failed requirement or skipping mandatory assurance. If

no compliant and workable option remains, escalate inability to meet the obligation through the established business process.

Protect cost predictability in the same agreement. Separate the funding ceiling from the current cost forecast, include review and operating effort, and identify who may authorize contingency spend. Model demand and usage costs under plausible scenarios. A task budget stops uncontrolled spending; it does not prove the whole service is affordable.

Manage dependencies as commitments between teams

Each dependency names a supplying owner, consuming owner, acceptance contract, required date or window, forecast, and escalation trigger. Include security and model validation capacity, procurement, infrastructure, test environments, data access, and change windows. A simulator permits earlier learning but does not mark the real dependency complete.

At program level, model the dependency network and common constraints. Teams using one vendor, review service, or environment have correlated risks. Do not multiply independent completion probabilities or average team confidence percentages to claim a program probability without a justified model. Provide scenarios when the network is not sufficiently understood.

A leadership update that supports a decision

Illustrative update, not a real project forecast. The bank's first slice is a supervised email-drafting service for one inquiry category. The team can commit to an integration decision on 23 October within the approved discovery budget. The current judgment-based pilot range is 9 to 20 November, assuming approved test data by 19 October and a review slot during the following week. External sending and automated financial decisions are excluded. The principal uncertainty is case-system access. If access is unconfirmed on 19 October, the sponsor must choose a read-only pilot or revise the release expectation. A simulator demonstration does not settle that decision.

Once historical evidence supports a probability forecast, replace the judgment label with the method and calibrated range. Keep the purpose, scope, quality, cost, remaining unknowns, and requested decision visible. Leadership receives a stable reporting contract even while the team learns quickly.

7 Roles and decision rights

Combine compatible work around an outcome. Preserve the expertise and decision rights that make the result dependable. These are **responsibility bundles**, not a requirement to create more job titles. Existing titles can remain while people acquire broader capabilities.

The responsibility bundles

Product and outcome owner. Owns business intent, benefit, priority, minimum useful scope, and user acceptance. Accepts business residual risk only within delegated authority. A product owner does not automatically acquire model-risk, data-use, or release authority.

Delivery integrator or forward deployed engineer FDE. Works close to users and combines enough analysis, design, coding, integration, evaluation, and coordination to carry a bounded outcome through the lifecycle. Keeps intent, dependencies, and evidence connected. Escalates beyond personal competence. The FDE is a delivery role; it is not a universal substitute for an architect, data scientist, domain expert, or independent reviewer.

AI workflow operator. Configures and supervises digital workers: approved context, tools, task decomposition, access, evaluation runs, budgets, monitoring, exceptions, and recovery. May build reusable workflows and diagnostic tooling. Delivery agents and production agents have separate

identities and mandates. This role needs operational and engineering judgment appropriate to its authority; operating a prompt interface alone is insufficient.

Engineering owner. Owns technical design, integration, maintainability, security implementation, and build integrity. Architects and engineers supply the relevant depth. An FDE may hold this responsibility for a system within demonstrated competence and organizational policy.

Assurance owner. Owns the evaluation approach, evidence sufficiency, challenge, and recommendation on acceptance. Works with testers, domain experts, researchers, security, data specialists, and independent model validation where required. AI can execute checks; accountable people judge what the evidence establishes.

Flow and forecast lead. Owns the integrity of the delivery forecast, dependency coordination, active-work limits, review-capacity planning, and timely escalation. The product owner owns scope choices and the sponsor authorizes commitments within policy. Project managers and scrum masters can combine complementary parts of this bundle, while program managers coordinate across outcomes and shared constraints.

Service owner. Owns live service performance, support, resilience, supplier dependencies, incident response, operational acceptance, and retirement. Platform, release, and operations engineers provide the implementation. Production AI operators work within this service accountability.

Where FDE and AI operator combine

In a narrow, limited-impact workflow, one skilled FDE may also operate the delivery agents and own engineering. Another person can provide product acceptance or peer assurance where policy permits. The work must still have named service cover and an independent route for consequential decisions. In a bank or insurer, use shared specialists and risk functions according to the use case; a small core team does not make those responsibilities disappear.

Separate the FDE and operator when concurrent workflows, production support, access sensitivity, or platform complexity exceed one person's capacity. Separate assurance when the builder cannot challenge the evidence credibly or policy requires independence. Do not consolidate roles merely to meet a headcount target.

Configuration	Responsibilities that may combine	Support that must remain available
Limited-impact internal workflow	FDE plus developer plus delivery-agent operator; product owner plus flow coordination	Peer challenge, named service owner and deputy, approved tooling, and specialist advice when needed.
Customer-facing bank or insurance service	FDE combines implementation and selected BA or solution-design work; project manager and team coach coordinate flow where competent	Dedicated or shared assurance, domain expertise, design and research, data and model specialists, and service cover.
Critical or highly autonomous service	Several FDEs integrate bounded parts; a separate operator manages significant agent workloads	Independent validation where required, separate authorized approvals, specialist architecture and security, production operations, and tested succession.

These are composition examples, not team-size targets. Combining activities does not mean one person must possess every discipline at equal depth.

Mapping traditional roles to the new work

The mapping describes plausible combinations, not a mandatory career destination or a finding that a role has become redundant.

Traditional role	Can combine into	Judgment and responsibility retained
Software developer	FDE, engineering owner, or AI workflow operator	Code and system understanding, integration, debugging, maintainability, and review of generated changes.
Tester	Assurance owner or evaluation specialist; may help operate test agents	Independent test design, failure exploration, evidence quality, and release challenge.
Business analyst BA	FDE with technical development, or product and domain analysis	Meaning of business rules, process exceptions, requirements, and acceptance examples.
Designer	FDE with design depth, or shared experience specialist	Interaction quality, accessibility, comprehensible approvals, and human recovery.
Researcher	Discovery and assurance specialist; may join an outcome team	Real-user evidence, study design, unmet needs, and validity of conclusions.
Project manager	Flow and forecast lead	Dependencies, capacity, risks, commitments, stakeholder decisions, and forecast history.
Architect	Engineering owner or shared architecture capability	System constraints, quality attributes, technical coherence, and long-term trade-offs.
Data scientist	Model and evaluation specialist; FDE where breadth permits	Model suitability, experimental design, uncertainty, performance, and limitations.
Data analyst	Outcome measurement and forecast analysis	Data definitions, lineage, baseline quality, cohort analysis, and interpretation.
Business architect	Product and outcome strategy across teams	Business capabilities, operating model, cross-process effects, and value alignment.
Solution architect	Engineering owner or FDE with architecture depth	End-to-end design, integrations, trust boundaries, nonfunctional needs, and recovery.
Release engineer	Platform and service engineering; AI workflow operations	Build provenance, deployment controls, environment integrity, rollback, and release evidence.
Product owner	Product and outcome owner	Priority, scope, value, acceptance, and communication of the product goal.
Scrum master	Team effectiveness and flow coaching; may contribute to the forecast bundle	Facilitation, impediment removal, sustainable work, and improvement of collaboration.
Program manager	Cross-team outcome and dependency coordination	Integrated roadmap, shared capacity, program scenarios, business readiness, and escalation.

If an organization continues to use Scrum, preserve its defined accountabilities and practices; a renamed combined role does not by itself constitute Scrum. [17] This framework also works with other delivery cadences.

Competence and capacity before consolidation

Create a responsibility map for each initiative. For every bundle name the primary person, deputy, demonstrated skills, decisions allowed, protected review time, specialist support, and escalation route. Practise a handover so shared knowledge can support someone other than the original FDE.

Deputies use their own approved identities and receive only the delegated authority required for cover. Shared credentials or the absence of an approver do not create permission. If qualified cover is unavailable, narrow activity or invoke the agreed continuity arrangement and update affected commitments.

Qualify broader roles through observed work: understanding a real workflow, identifying ambiguous requirements, producing and reviewing a bounded change, evaluating failures, enforcing tool limits, diagnosing integration issues, and recovering a failed run. Start at limited authority; use coached reviews and supervised practice before expansion. Preserve specialist development paths and opportunities for junior staff to learn rather than only checking AI output.

Reduce active work when one person becomes the bottleneck for discovery, agent supervision, review, and operations. Measure interruptions and review age before adding more agents. Having fewer handoffs can improve continuity, but concentrating knowledge and authority in one person creates absence and oversight risks.

Decision rights and separation

Decision	Accountable decision maker	Required input
Fund or stop the initiative	Business owner	Delivery, service, risk, and user evidence
Accept scope and intended behaviour	Product owner	Domain users, engineering, assurance
Authorize technical design	Engineering owner	Service and applicable specialists
Approve data use	Data owner	Privacy, security, and contractual requirements
Accept evaluation method and results	Assurance owner	Domain experts and engineering
Authorize release within policy	Named release authority	Product, engineering, assurance, service, and any required risk approval
Contain a live incident	Service owner or delegated incident commander	Engineering, security, domain experts
Expand agent authority	Relevant business authority owner	Engineering enforcement and assurance evidence

Digital workers may draft recommendations and evidence for all these decisions. Record the human decision, exact version reviewed, conditions, and date. Unanswered approval requests remain pending; elapsed time is never approval.

For material and critical changes, the same person must not both produce the work and provide its required independent approval. An operator cannot expand their own mandate; a builder cannot weaken release criteria to pass. Data approval, model validation, business acceptance, and release authority remain with the roles delegated by policy. A second agent is not a substitute for independence.

Human readiness is part of system readiness

A named reviewer needs domain competence, usable evidence, enough time, and authority to reject or pause work. Before release, test whether reviewers can identify representative errors, explain a decision, and complete escalation under expected workload. Define a queue limit and a fallback when review capacity is exceeded. A nominal human checkpoint without these conditions does not satisfy readiness.

Develop judgment through coached case reviews: establish the goal, examine the evidence and current reality, explore options, and agree the next action. This adapts the GROW coaching structure discussed in the posts. [12] Preserve direct instruction for urgent containment. Encourage challenge and near-miss reporting; review the conditions that produced a mistake as well as individual actions.

8 How teams work together

Organize small **outcome teams** around a user journey or service slice. Each team has access to product, integration, engineering, assurance, forecasting, and service responsibilities, whether dedicated or shared. Organize shared capabilities around reusable services and scarce expertise. This is a collaboration design, not a fixed staffing ratio.

How teams connect

Team or function	Provides to the outcome team	Agreement needed
Business and domain users	Intent, rules, representative cases, acceptance, and operational context	Named decision owner, response time, decision scope, and feedback access.
Platform and AI enablement	Approved tools, identities, environments, reusable workflows, logging, and evaluation services	Service interfaces, capacity, support, change notice, and recovery.
Data and model specialists	Data access, lineage, quality, model analysis, and specialist evaluations	Permitted use, delivery dependency, versioned datasets, limitations, and review route.
Architecture, security, privacy, and accessibility	Constraints, reusable patterns, design advice, and specialist evidence	Early engagement, reserved review capacity, acceptance criteria, and escalation.
Risk and independent validation	Challenge and required assurance decisions	Evidence expected, independence, decision authority, and realistic review windows.

Team or function	Provides to the outcome team	Agreement needed
Operations and support	Live readiness, continuity, customer recovery, and incident ownership	Runbooks, exposure limits, support coverage, handover evidence, and stop authority.
Program and portfolio coordination	Cross-team priority, integrated forecasts, funding, and conflict resolution	Outcome ownership, shared dependencies, capacity allocation, and decision deadlines.

A slice moving through the organization

The product owner and domain users agree the result and examples. The FDE and researcher test important assumptions; the designer checks the interaction. Engineering, data, architecture, and assurance establish the design and evidence approach before significant build effort. The operator runs bounded agent tasks and supplies traceable results. The FDE integrates the slice; assurance challenges it against independent cases. The service owner accepts operational readiness, and the authorized release owner makes the release decision. Users and operations supply observed outcomes that change the backlog and forecast.

People collaborate on the same slice where useful instead of handing completed documents down a chain. Each receiving function must still understand and accept what it receives. AI-generated records can support that exchange; they cannot substitute for a consequential decision.

Reusable controls without a central queue for every change

The platform team supplies approved deployment patterns, authorization components, environment controls, worker profiles, and evaluation infrastructure. Specialist functions publish evidence requirements and change routes. Teams reuse current evidence only when scope and assumptions remain applicable. Novel use, expanded authority, material data changes, and failed controls take an explicit exception or renewed-review path.

Agree service expectations with shared review teams and measure their actual queue and completion times. Book scarce specialists against the integrated plan. Escalate capacity conflicts to the program or portfolio owner with the effect on scope, date, cost, and risk. Silence is not approval; a review target is not an automatic waiver.

Preserve organizational challenge

Within the organization's governance model, delivery and service teams own the risks they create and operate their controls. Risk, compliance, and independent model validation provide the challenge and approvals assigned to them. Internal audit can assess the framework and its operation without becoming the routine release approver. Do not relabel a peer inside the delivery team as independent validation where organizational policy requires a separate function.

For banks, connect the delivery dependency map to critical operations and their continuity plans. BCBS guidance emphasizes resilience, interdependencies, third parties, and response to disruption. [19] For insurers, include customer explanation, fair treatment, and redress in the acceptance and operating design; the IAIS AI application paper addresses these alongside governance and robustness. [20] The institution must determine local applicability and required approvals.

Make disagreements actionable

If product wants a broader scope than evidence supports, assurance records the gap and the release authority resolves exposure within policy. If a dependency changes, the supplying owner informs consuming teams and the forecast lead updates scenarios. If a live control fails, the delegated incident commander contains it immediately under the incident plan. Budget or date pressure does not empower the FDE to override an independent stop decision.

9 The team digital brain

The brain is a governed knowledge system, not an undifferentiated transcript archive. It can connect a knowledge workspace, work tracker, code repository, artifact store, model registry, and operational dashboards. Keep an authoritative location for each artifact and link it, rather than copying the same content into multiple competing records.

Knowledge structure

Maintain linked records for outcomes, source evidence, requirements, assumptions, decisions, risks, architectures, datasets, models, prompts, tool contracts, tests, evaluations, releases, incidents, feedback, forecasts, uncertainty records, dependency agreements, and lessons. Each approved release should be traceable through:

Outcome → source and assumption → requirement → design → implementation → test or evaluation → release → observed result.

Link forecast snapshots to the scope and evidence available on that date. When an assumption changes, notify the owners of affected requirements, dependencies, releases, and commitments. The shared record makes the effect of learning visible across teams.

Each record includes an identifier, owner, status, source links, creation and review dates, access classification, version, related records, and retention rule. Separate observed evidence from AI interpretation. A summary must link to its source and identify unresolved ambiguity.

Use the states **captured, proposed, reviewed, approved, superseded, and retired**. Record what supersedes an old decision. Search should prefer current approved material while allowing authorized access to history. A later timestamp alone does not make a statement authoritative.

Capture and review

For suitable meetings, use approved recording and transcription tools, with required notices and permissions. For sensitive conversations or unavailable recording, capture reviewed notes and decisions. The method requires reliable records, not recording in every circumstance. Do not move confidential material into unapproved personal AI tools.

AI extracts candidate decisions, requirements, assumptions, dependencies, and actions. A designated reviewer checks meaning and asks participants to resolve consequential ambiguity. Do not turn tentative discussion into commitment or attribute a decision to an uncertain speaker. Preserve corrections alongside the original source.

For a decision meeting, state the decision needed, its owner, and the evidence required in advance. Close with the decision or unresolved question, rationale, action owners, and dates; confirm these with participants. Link the record to the affected requirement or risk and check follow-through at the next review. This makes capture serve decisions and learning. [8]

Controlled context for digital workers

Give each worker a task-specific context pack: approved requirements, applicable standards, relevant design, permitted data, test contract, and known exclusions. Include source versions and unresolved questions. Do not provide the entire organizational archive by default.

Apply source permissions at retrieval time and propagate revocation to indexes, caches, and derived views. Treat retrieved text, emails, uploaded files, and tool responses as data that cannot grant new permissions. Screen external contributions before promoting them into approved knowledge. Test stale retrieval, conflicting instructions, malicious content, and cross-user leakage.

At each accepted cycle, propose targeted updates to the relevant product, engineering, security, data, and operations records. Link each update to the changed artifact and evidence; preserve the earlier version. A failed or unfinished attempt belongs in the learning record, not in the approved description of current behaviour. This adapts PACE's maintained context documents to the wider team digital brain. [23]

If a context refresh fails, identify the affected tasks. Pause work that depends on a changed permission, critical rule, or unverified system description; unaffected work may continue using explicitly verified sources. A generated summary cannot approve a policy change.

Assign a knowledge steward to review stale records and unresolved contradictions weekly. Measure the proportion of active requirements with current sources, ownership, and evidence links. The brain improves delivery only if people can trust what it says and find what they need.

10 The end to end lifecycle

The stages form a continuous learning loop. Work can overlap and return to earlier decisions when evidence changes. The gates below are evidence checkpoints, which may be asynchronous or automated under approved policy. They do not require a separate committee meeting for every stage. For each slice, identify which uncertainty the next activity reduces and what new evidence changes the decision. Reuse current approved evidence where justified; revisit affected gates when assumptions change.

Stage 0 Establish the delivery conditions

Purpose. Make the team able to deliver and operate responsibly. The business owner and delivery integrator establish scope, decision rights, approved AI tooling, environments, repository controls, specialist access, and the initial digital brain.

Outputs. Initiative passport; D/P/I classification; impact and authority assessment; team charter; tool and supplier register; initial budget; ownership of operational support.

Actual-use check. Ask team leads which AI tools were used in the last 30 days, including personal or free accounts, browser extensions, and AI features in existing software. Record purpose, users, information shared, owner, and approval status. Reconcile actual use with the approved register; assess unfamiliar uses and assign remediation or an approved alternative. [10] Use a proportionate disclosure process within organizational policy.

Uncertainty and predictability. Can this team obtain the access, competence, review capacity, and operating support it needs? Record unresolved conditions in the uncertainty and dependency registers. Commit to a bounded discovery effort, not an unsupported production date.

G0 Ready to investigate. A sponsor and service owner are named, tools and data access are appropriate, and the team has a bounded discovery budget. Missing permissions or environments become explicit dependencies with owners rather than hidden waiting time.

Stage 1 Discover the problem and establish a baseline

Purpose. Understand the user task and decide whether AI is useful. Observe the current workflow, capture user evidence, identify affected people, measure current cost and quality, and compare doing nothing, process change, conventional software, purchased capability, and AI options.

AI contribution. Summarize approved research, cluster evidence, identify contradictions, and suggest experiments. Humans validate the problem and important interpretations.

Explore, then select. First invite alternatives and challenge the problem framing with users and affected teams. Then compare options using value, evidence, cost, risk, and feasibility; choose a bounded experiment and owner. Label the two modes so early criticism does not close exploration. AI can propose possibilities, but generated personas do not replace affected people's perspectives. [7] Check the whole workflow: faster drafting may move effort into review, support, or another team's queue. [9]

Outputs. Outcome brief; user journeys; current baseline; constraints; alternatives; initial impact assessment; experiment plan.

Uncertainty and predictability. Is the problem real and is the proposed benefit plausible? Use user evidence and a baseline to narrow the outcome. Update the range of solution and delivery scenarios; retire options that the evidence does not support.

G1 Worth pursuing. The product owner accepts a measurable outcome, a credible benefit hypothesis, and a reason to select the proposed approach. Stop if the need is weak, prohibited, or infeasible under known constraints.

Stage 2 Define behaviour and acceptance

Purpose. Translate intent into small, testable slices. Define happy paths, exceptions, failure handling, accessibility, privacy, security, latency, cost, and operational requirements. For AI outputs, specify what an acceptable range of behaviour is, including when to ask, abstain, or escalate.

AI contribution. Draft stories and scenarios from reviewed sources, surface gaps, and propose tests. Domain users validate semantics and consequential edge cases.

Outputs. Prioritized backlog; behavioural contracts; initial evaluation dataset and rubric; acceptance thresholds; prohibited actions; assumptions and dependencies.

Uncertainty and predictability. Do people agree what acceptable behaviour means? Turn ambiguity into examples and measurable limits. Record the minimum useful scope and what can change when a date is fixed.

G2 Behaviour and acceptance agreed. Product and assurance owners agree what will be measured and what blocks release. Unresolved high-impact ambiguity cannot be delegated to a model to guess.

Stage 3 Design the system and its control boundaries

Purpose. Design the full workflow, including humans, models, tools, data, and recovery. Decide where deterministic logic is required, whether retrieval or fine-tuning is justified, and how the service operates during model failure. Compare supplier options against the task, not generic benchmark position.

Outputs. Architecture and data-flow diagrams; threat and failure analysis; interaction contract; model selection record; supplier assessment; service targets; authorization design; fallback and rollback plan.

Uncertainty and predictability. Can the proposed design enforce the required limits and meet the operating needs? Test the highest-consequence technical assumption early. A new dependency or architectural constraint can widen the delivery range.

G3 Design accepted. The engineering owner and applicable reviewers confirm that access and limits are enforceable, critical failures have an owner and response, and the design can be tested and operated.

Stage 4 Prepare data and the evaluation environment

Purpose. Establish usable, permitted data and realistic test conditions. Verify data rights, collection purpose, quality, coverage, labels, lineage, retention, and access. Build integrations, sandbox tools, simulation, and representative datasets before relying on demonstrations.

Outputs. Data and model records; isolated development, validation, and held-out test sets; annotation guidance; simulator; environment and integration contracts; non-AI baseline.

Uncertainty and predictability. Are the data and test environment representative enough for the next decision? Distinguish simulator evidence from real integration evidence. Forecast any remaining access, validation, or environment dependency explicitly.

G4 Ready to produce evidence. Data and assurance owners confirm permitted use, sufficient coverage for the next experiment, separation of test material, and an environment that can safely expose meaningful failures. Document gaps that require later live validation.

Stage 5 Build and integrate in small increments

Purpose. Implement approved slices using people and digital workers. Workers receive a task contract, scoped permissions, budget, context, expected output, and stopping rule. Humans resolve ambiguity and review consequential changes.

Outputs. Versioned code, configuration, infrastructure, prompts, models or adapters, tests, documentation, dependency records, and deployment artifacts. All meaningful behaviour changes link to requirements and risks.

Uncertainty and predictability. Can the slice be integrated with acceptable rework? Track actual completion, blocked time, review age, and new defects. Limit concurrent agent tasks to available integration and assurance capacity.

G5 Candidate ready for independent verification. The engineering owner confirms review, reproducible build inputs where feasible, passing basic checks, and evidence that the candidate implements the approved scope. Generated code and tests are subject to the same standards as human contributions.

Stage 6 Verify the system and validate the outcome

Purpose. Test software correctness and evaluate behaviour, task success, harms, and recovery. Use held-out cases, adversarial cases, representative users, repeated trials where outputs vary, and integration tests with real dependencies where available.

Outputs. Test and evaluation report; results by relevant segment; uncertainty and limitations; user acceptance evidence; security findings; residual risks; release recommendation.

Uncertainty and predictability. How confidently does the candidate meet acceptance criteria for the intended users? Report sample limits, variation, segment failures, and remaining exclusions. A quality failure changes the release forecast or scope; it never changes the pass rule silently.

G6 Candidate acceptable. The assurance owner accepts the evidence, the product owner accepts task outcomes, and required risk owners accept residual risk. Failed hard constraints block progression. Averages cannot hide critical or poorly served segments.

Stage 7 Release under controlled exposure

Purpose. Deploy an identified release with an explicit exposure and authority limit. Validate access, configuration, support, monitoring, capacity, fallback, and recovery. Use shadow evaluation, a limited cohort, or a canary where suitable for the workflow.

Outputs. Release manifest; signed decision record; deployment plan; user communication and support guidance; live scorecard; rollback rehearsal; approved autonomy mandate.

Uncertainty and predictability. Can the identified version operate within the approved exposure? Resolve required readiness gaps, confirm release-window dependencies, and record the commitment separately from the authorization decision.

G7 Authorized for the stated exposure. The release authority confirms the evidence and the service owner accepts readiness, including reviewer capacity, supported product claims, and the response rehearsal appropriate to impact. Conditions state who may use the release, what it may do, and what stops rollout. Emergency changes use a documented fast path and subsequent review.

Stage 8 Operate and learn

Purpose. Monitor real task outcomes alongside technical health. Track incorrect outputs, failed or unauthorized actions, user corrections, retrieval problems, unusual usage, costs, supplier changes, and shifts in the incoming workload. Sample quality even when users do not complain.

Outputs. Operational scorecard; incident and near-miss records; sampled quality review; feedback linked to requirements; updated risk assessment; benefit evidence.

Uncertainty and predictability. Does the live service sustain the expected benefit and behaviour? Compare operational distributions with the evaluated baseline. Drift or changed demand can reopen previously bounded uncertainty and trigger restriction or re-evaluation.

G8 Continue, restrict, or stop. The service owner reviews evidence at the agreed cadence and on trigger events. Critical issues invoke immediate containment. Sustained deterioration causes narrowed scope, additional human review, rollback, or suspension.

Stage 9 Improve or retire

Purpose. Convert lessons into controlled changes or end the service. Evaluate proposed improvements against the baseline and current version. Route changes through the relevant earlier gates. Periodically reconsider whether the service is still useful, affordable, and supportable.

Outputs. Improvement decisions and new evaluation cases; or migration plan, communication, data disposition, revoked access, retired endpoints, and retained evidence.

Uncertainty and predictability. Is continued investment justified, and can change or retirement be completed safely? Estimate migration and downstream effects, verify closure evidence, and retain ownership of residual obligations.

G9 Change or closure accepted. The designated owner approves a new release route or confirms retirement. Decommissioning includes model access, agent credentials, schedules, indexes, caches, integrations, and downstream consumers. Keep required records and remove data according to approved retention rules.

How to move without recreating waterfall

An experiment can revisit discovery while another accepted slice is being built. A familiar change may reuse the architecture and data evidence, then repeat the affected checks. A new purpose or expanded

authority returns to classification and design. The gate record names the evidence reused and the reason. No route bypasses an applicable hard constraint.

Working definitions

Ready for implementation means the outcome, owner, sources, acceptance method, access, dependencies, and risk route are clear enough to execute a bounded task. **Done for release** means the candidate passed its required checks, decisions are recorded, recovery is tested, and operational ownership is accepted. **Outcome achieved** means production evidence meets the business target over the specified observation period. These are three different states.

11 Engineering with digital workers

Use a worker registry with each worker's purpose, owner, model or service, input sources, permitted tools, accessible environments, data restrictions, budget, logging, and escalation route. Keep delivery-worker credentials separate from product runtime credentials. Production access must be explicitly justified and constrained.

A typical pipeline has an analyst worker that drafts specifications, a builder that implements them, a test designer that derives cases from approved requirements, an executor that runs checks, and a reviewer that identifies problems. These are logical responsibilities; they need not be five separate agents or run concurrently.

Independence comes from the evidence. Two agents can share the same mistaken assumption. Keep acceptance criteria and held-out tests outside the builder's control. Use deterministic assertions, independent domain judgments, real system observations, and suitable human review. A model reviewing its own output is useful feedback, not sufficient release evidence.

For every task, supply objective, non-goals, artifact boundaries, approved context, permissions, acceptance tests, resource limits, expected evidence, and stop conditions. Workers stop for ambiguity that changes scope or consequences, failed permissions, repeated unsuccessful attempts, unexpected sensitive data, or exceeded budgets. They cannot approve their own privilege increase, alter required acceptance criteria to pass, or bypass a failed gate.

Protect repositories and builds with peer review, dependency and license checks, secret scanning, vulnerability checks, isolated execution, and approved package sources. Keep records sufficient to explain significant AI-assisted decisions without retaining unnecessary sensitive prompt content. The exact generated text may not be reproducible, but the accepted artifact and its verification evidence must be identifiable.

NIST SP 800-218A extends secure development guidance for AI model and system producers and acquirers. Use that guidance alongside the organization's secure engineering standard; the worker pipeline above is this framework's operating design. [4]

Make handoffs verifiable

Use a small, structured record between planning, implementation, testing, security, operations, and knowledge maintenance. Link the cycle identifier, approved contract version, candidate manifest, changed artifacts, executed checks and raw evidence, remaining findings, and next owner. Distinguish a worker's statement from a tool-observed result. PACE's daily artifact chain is a useful implementation example; this framework does not require its six-agent arrangement. [21]

Required checks must be enforced outside the model. Record test exit status, tests collected and executed, failures, skips, environment, and the exact candidate tested. Missing evidence, a timeout, or an unrun mandatory check cannot become a pass because an AI report says so. Keep full logs in

controlled storage and give reviewers useful summaries with links. Protect check definitions and acceptance rules from the worker being assessed.

Record every attempt, including failed and abandoned runs, with elapsed time, cost, tool actions, reason, and resulting state. On resumption, verify the candidate, contract, context, permissions, and external state before reusing evidence. Reuse unaffected checks; repeat invalidated checks. A completion file alone is insufficient proof that a changed candidate is accepted. Preserve action identifiers and receipts so retries cannot duplicate external actions.

Manage findings without hiding debt

Separate release blockers from findings that an authorized owner may defer within policy. Each deferred finding needs a stable identifier, severity and exposure, owner, rationale, due date, interim control, and evidence required for closure. Review overdue findings at each cycle; reserve capacity to resolve them. Reassess immediately when exposure changes. This extends PACE's advisory backlog idea. [25]

Close findings individually against evidence for the current candidate. A general approval of an agent report cannot clear unrelated items. Mandatory safety, security, privacy, or acceptance requirements remain blockers; relabeling them as advisory or out of scope does not remove them. Human intervention is triggered by consequence, uncertainty, mandate changes, and failures, as well as planned reviews. [24]

Reduce avoidable execution variation

Use versioned worker profiles, approved context packs, known tool interfaces, and repeatable execution environments for recurring tasks. Record the accepted artifact and evidence, rather than expecting identical generated wording. Add a regression case after a repeated failure; change the task contract or workflow when it caused the failure. Repeated prompting without diagnosing the cause is rework, not learning.

Pause new agent work when review capacity is full. Investigate long-running tasks and repeated retries; set task-level time, cost, and attempt limits. Use independent verification at the appropriate risk level. More parallel workers increase throughput only when downstream capacity and dependencies can absorb their results.

12 Engineering AI behaviour and data

Specify a behavioural contract

For each AI capability, define intended task and users; allowed input and output types; authoritative information sources; acceptable variation; quality thresholds; excluded uses; escalation or abstention conditions; maximum latency and cost; tool permissions; and the actions that must never occur.

For factual generation, evaluate support for claims and whether cited sources actually justify them. For prediction or ranking, assess suitable error measures, calibration, thresholds, and consequences across relevant populations. For agents, evaluate completed tasks, allowed actions, intermediate tool use, and resulting system state. A plausible final answer can conceal an incorrect action.

Choose and manage the model

Compare candidate models against representative cases, operational constraints, data terms, geographic requirements, service reliability, change notification, cost, and portability. Prefer the

simplest approach that meets the task. Document why rules, conventional software, retrieval, a hosted model, self-hosting, fine-tuning, or training is appropriate.

When training or fine-tuning, record dataset versions, annotation rules, training configuration, model artifacts, selection criteria, and evaluation results. Separate examples by user, entity, time, or other leakage boundary as appropriate. Track feature and label quality and possible mismatch between training and serving. A model being accurate on historical data does not establish fitness for a new context.

When using retrieval, test source authority, freshness, permissions, coverage, retrieval quality, and response faithfulness separately. Indexing and chunking changes can alter behaviour even when code and the model stay unchanged. When memory is used, define what may be stored, who owns it, when it expires, how users correct it, and how cross-session and cross-user isolation works.

Version the complete behaviour

The release manifest identifies application and infrastructure versions, model and provider configuration, prompts, policies, tool schemas, retrieval and embedding settings, corpus or dataset versions, memory policy, feature flags, evaluation suite, and approvals. Store secrets outside the manifest.

Prefer stable model versions when available. If a provider cannot pin behaviour, record that dependency and use recurring canaries and regression checks to detect changes. A supplier change is a system change for assurance purposes even if no internal code changed.

Production learning updates the backlog first. Data, prompt, policy, memory-rule, and model changes follow an approved validation route. Online adaptation is permitted only within a separately assessed and monitored adaptation policy. Do not automatically train on user feedback or retain it indefinitely.

Engineer dependable boundaries around variable components

Define which behaviour must be invariant and which may vary. Amount calculations, resource permissions, recipient scope, and committed state changes should use controlled logic with explicit checks. Wording and task strategy may vary within a behaviour contract. Lower randomness settings alone do not establish safety, factual accuracy, or repeatability.

Measure variation across inputs, repeated runs, user groups, model versions, and operating conditions. Set task-specific thresholds for quality, latency, cost, and escalation. Where the system cannot support a consequential case reliably, narrow its mandate, obtain human judgment, or use a suitable conventional path. A stable fallback is part of predictable service.

13 Testing and evaluation

Use a portfolio of tests. Software tests check specified behaviour; AI evaluations estimate performance over a defined distribution of cases. Both are necessary when a product contains AI.

Evidence layer	What it establishes
Conventional engineering tests	Units, integration, APIs, data integrity, migrations, permissions, accessibility, performance, and recovery behave as specified.
Model and retrieval evaluation	Task quality, factual support, relevance, calibrated prediction, abstention, and data access behave acceptably.

Evidence layer	What it establishes
Workflow and agent evaluation	The end-to-end task is completed, tools are used correctly, retries do not duplicate actions, and resulting state is correct.
Human interaction evaluation	Users understand limits, make effective approvals, recover from errors, and can use an accessible alternative.
Adversarial and failure testing	Malicious inputs, unavailable tools, poisoned content, ambiguity, privilege abuse, and resource exhaustion meet containment expectations.
Production validation	Actual users and live operating conditions sustain quality, benefit, cost, and reliability.

Dataset and judgment rules

Maintain development cases for iteration, validation cases for selection, and a held-out release set protected from tuning. Include permissioned historical cases, expert-authored cases, boundary conditions, adversarial cases, and synthetic cases. Preserve provenance and avoid splitting near-duplicates across sets. Refresh held-out sets when repeated exposure compromises their value.

Synthetic personas are useful for widening scenarios, but they do not prove real user behaviour or population fairness. Validate their coverage against observed workflows. Label cases with a rubric; double-review consequential cases and adjudicate disagreements. When using an AI judge, measure its agreement with expert judgments on a suitable sample and examine disagreements, bias, and position effects. Do not let an unvalidated judge be the sole assessor of its own model's output.

For variable outputs, report repeated runs, configuration, success rates, sample sizes, relevant segments, and uncertainty. Do not repeatedly run a failing candidate and select a favourable attempt. A fixed seed can aid diagnosis but does not guarantee identical behaviour across providers or system changes.

Acceptance rules

Define hard constraints separately from statistical quality targets. Examples of hard constraints are no action outside authenticated scope, no unapproved recipient, no duplicate committed transaction on a retry, and no critical unresolved vulnerability. Any observed breach of a release-blocking constraint fails the candidate. Passing a finite test set does not prove that the true breach rate is zero.

Quality criteria specify a metric, segment, threshold, sample plan, evaluator, and decision rule before testing. For example, the business owner might require that the lower confidence bound for correct routing exceeds an agreed threshold, with stricter handling for sensitive categories. The threshold must come from the use case and consequences; there is no universal acceptable AI accuracy percentage.

Evaluate against a conventional or current-service baseline. Include latency, cost, human correction effort, and harm severity. Record unsupported segments and exclude them from launch until evidence is sufficient. A favourable overall average is insufficient when a critical workflow fails.

NIST's Generative AI Profile provides lifecycle risk considerations including pre-deployment testing, provenance, and incident disclosure. This playbook translates those themes into the operating practices above; it is not a NIST-prescribed test suite. [3]

Test variation and measure the evidence itself

Evaluate routine conditions and plausible tails: high workload, incomplete information, rare but serious errors, and delayed dependencies. Fix the sampling plan and scoring rules before comparing candidates. Separate observed stability of an output from correctness and useful task completion. Report variation in human judgments and resolve ambiguous labels before interpreting small model differences.

Quality confidence and schedule confidence are different measures. An 80 percent forecast of delivery by a date says nothing about acceptable model accuracy. A passed evaluation supports a defined scope and version; it does not guarantee future behaviour under a changed supplier or population.

14 Users interacting through AI

Support three interaction patterns explicitly: a person using the conventional interface, a person using the product's assistant, and an external AI acting for that person. All routes should reach consistent business rules and access checks.

Define an interaction contract with stable task and resource identifiers, clear tool or API schemas, permissions, errors, rate limits, asynchronous job status, retry semantics, and receipts. Where browser interaction is supported, use accessible controls, semantic structure, visible state, and clear success or failure feedback. A dedicated API is preferable where the product needs dependable machine interaction, but it does not replace user-facing accessibility.

Delegation and confirmation

Bind agent actions to an authenticated principal and, where applicable, a verifiable delegation. Check that the person is entitled to the resource, that the agent's mandate permits the action, and that current policy allows it. Use short-lived, revocable credentials with resource and action scope. An agent's assertion of identity or a message saying that a manager approved an action is not authorization.

For material commitments, show the exact target, relevant data, recipients, amount or scope, and consequence before approval unless an approved bounded mandate already covers the action. Bind approval to that specific proposal, with expiry and revalidation when the proposal changes. Prevent a stale approval from authorizing a changed transaction.

Enforce policy and validate inputs outside the model. Use idempotency keys, duplicate detection, transaction boundaries, and compensating actions where appropriate. Limit repeated calls, total spend, delegation depth, and task duration. Return a receipt showing what actually happened and what remains pending; do not claim success merely because a tool was invoked.

Human control and security

Provide clarification, cancel, undo where feasible, and human escalation. Preserve relevant context during handover without exposing unnecessary data. Explain whether the system is advising, drafting, or executing. Evaluate whether users can understand and use confirmations under real workload, rather than assuming a confirmation button supplies meaningful oversight.

Test indirect instructions inside emails, documents, web content, and tool results; poisoned memory; credential misuse; altered tool schemas; hostile outputs; and cross-tenant access attempts. These are useful test themes reflected in OWASP's agentic security guidance. [5] Also test legitimate mistakes, language variation, inaccessible interactions, agent retries, delegated authority expiry, and revocation during an in-flight workflow.

Make outcomes understandable across interaction routes

Keep business rules consistent whether a person clicks, a built-in assistant acts, or an external AI uses an interface. Expose clear pending, completed, failed, and cancelled states. Define what happens when a user changes their mind, an approval expires, or an agent disconnects mid-task. A receipt and a reconcilable record reduce uncertainty about the action actually taken.

15 Release control and operational response

Route changes according to what they can affect

Change	Default evidence route
Editorial documentation with no behaviour effect	Peer review and documentation checks.
Conventional code or dependency change	Impact analysis, targeted regression, security checks, and relevant release gates.
Prompt, model, tool, retrieval, corpus, or memory-rule change	Relevant behavioural evaluation plus regression of critical tasks and policy boundaries.
New purpose, population, sensitive data, tool permission, or autonomy	Renew classification, design and impact review; repeat affected evaluation and release approval.
Emergency fix	Constrained authority, minimum critical checks, explicit exposure limit, and time-bound retrospective validation.

The engineering and assurance owners record why unaffected evidence can be reused. Approval is tied to a version and operating scope. Expired exceptions or materially different configurations invalidate the relevant approval.

Operate with explicit triggers

Each live capability has an owner, service and quality targets, cost budget, dashboard, alert thresholds, escalation route, and tested fallback. Observe task traces and important state changes with redaction and restricted access. Avoid indiscriminate storage of full prompts, secrets, or personal information.

For each critical control, link the risk, enforcing mechanism, evidence or signal, accountable owner and delegate, trigger, and required action. Confirm that someone can act on the signal. This connects data, model, security, and operational responsibilities across team boundaries. [14]

Monitor business success, sampled correctness, false positive and false negative patterns, unsupported claims, refusal and escalation rates, user corrections, tool failures, blocked actions, abnormal loops, latency, cost, data freshness, and provider behaviour. Look for changes in input distribution as well as outcome quality; workload change alone does not establish performance degradation.

Set response rules before launch. Examples: a confirmed unauthorized action stops the affected execution path; loss of an essential source routes work to a human queue; a sustained quality breach restricts the affected cohort; a cost spike enforces a task budget and opens investigation. Choose numeric thresholds, duration, and ownership for each service rather than importing arbitrary universal limits.

Verify the promises made to users

Record material claims in the interface, demonstrations, sales material, and operating guidance. Link each claim to an owner, evidence, the evaluated version and population, limitations, and a review trigger. An observed pilot benefit must not become an unsupported promise for every customer or workflow. Review claims at G7 and when production evidence changes; narrow or remove claims that no longer hold. This operationalizes the post's connection between trust, evidence, and accountability. [13]

Incident playbook

1. Detect and classify the problem using impact and scope.
2. Contain it by disabling a tool or capability, revoking credentials, limiting traffic, or switching to a tested manual or conventional path.
3. Preserve relevant versions, action receipts, and evidence with appropriate access and retention.
4. Identify affected users, records, commitments, and downstream systems; follow applicable notification and escalation procedures.
5. Correct state and recover service. Rolling back code does not undo messages, transactions, disclosures, or decisions already made.
6. Investigate causes across requirements, data, prompts, tools, models, policies, and human review; add regression cases and verify the fix before restoring authority.

Also rehearse model-provider outage, tool outage, data-source corruption, credentials compromise, and emergency switch-off. Log near misses and user corrections as learning evidence. Retirement follows the same discipline of controlled access removal and verified downstream effects.

Leadership response rehearsal. Before an R2 or R3 launch, walk through a credible harmful outcome with product, service, assurance, and relevant specialists: for example, the AI links a customer to the wrong record and sends an incorrect message. Demonstrate who detects the issue, who can pause the capability, how affected records and people are found, who communicates, and who authorizes recovery. Record gaps, owners, and closure evidence; unresolved critical response gaps block G7. For R1 work, a short owner-and-peer walkthrough is sufficient. Repeat after material changes to authority or the response chain. This turns the accountability question in the governance post into an observable readiness check. [15]

Renew assurance when behaviour changes

Record the baseline and tolerance for each monitored outcome, the minimum observation window where relevant, and immediate-stop events. Compare a changed model, prompt, corpus, tool, or mandate with that baseline before expanding exposure. Do not average away a serious failure in a small customer segment. A provider change can invalidate evidence even when the application has no code release.

Keep operational continuity separate from reverting a version: identify and correct downstream effects already produced. Reassess the delivery forecast for remediation, and give affected business owners explicit recovery choices and limitations.

16 Cadence and measures

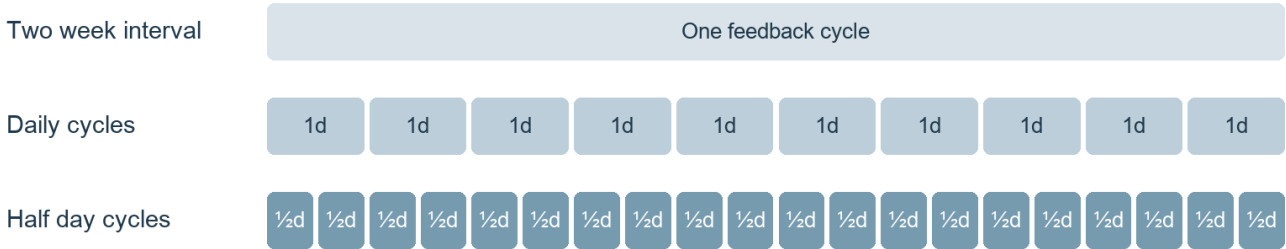
The organizing rhythm is a short, complete evidence loop. For suitable work, aim to move from a multiweek feedback interval to daily feedback, then to half-day feedback as preparation, automation,

and review improve. This is a design direction to test, not a universal speed claim. Teams already using continuous flow can improve their existing loop. Scrum does not require a two-week Sprint. [17]

Keep four clocks visible: **execution** is how long the work takes; **feedback** is when a result can change a decision; **release** is when an authorized candidate reaches users; **outcome** is when sufficient use demonstrates benefit or harm. They may run at different speeds. Faster generation improves delivery when integration, review, decisions, and observation can use the result.

Shorter intervals to verified feedback

Illustrative choices across ten working days



Every cycle closes with evidence and a decision. Longer checks remain pending.

Release authorization and observation of business outcomes keep their own timing.

Define a cycle around a decision

Choose one coherent, bounded outcome or important unknown. Agree the evidence required and the decision it will support before work starts. A cycle may produce an accepted slice, a rejected hypothesis, or evidence that a dependency must change. A learning result counts as learning; it is not reported as a released feature or achieved benefit.

Every cycle follows **Learn** → **Act** → **Prove** → **Sharpen**: understand the objective and evidence, perform focused work, test against agreed expectations, and use feedback to improve the solution and next lap. It can involve discovery, design, implementation, evaluation, or operations. AI used in delivery, AI inside the product, and AI acting for users retain their separate controls.

At the boundary record **accepted**, **needs revision**, **blocked**, or **stopped**, with the reason, evidence, remaining exposure, owner, and next action. Unfinished work retains its original identity and age when continued. The timebox ends even if the work is incomplete; neither acceptance nor a release is automatic. Replan the next cycle deliberately.

Prepare for a one day or half day cycle

Start when the team has a reviewed objective and acceptance contract, sufficient current context, accessible dependencies and test environments, a feasible evidence plan, a named decision owner, and review capacity within the window. Set the active-work limit, time and cost ceilings, retry limit, and stop conditions. Use existing approved patterns and controls to make this preparation lightweight.

If the slice is too large, split it into useful, independently verifiable parts or choose a learning experiment. PACE's proactive scoping is a useful prompt to inspect size before execution. [22] Here, AI may propose a split; the accountable owner approves any change to the contract or commitment. Do not cut mandatory controls, count acceptance criteria as a measure of complexity, or report the reduced scope as the original scope completed.

Illustrative operating windows

These schedules are examples for a team with ready work and available reviewers. They are not measured productivity benchmarks. Preparation, waiting, rework, and subsequent outcome observation remain visible in total lead time.

Activity	One day example	Half day example
Learn: confirm the objective, evidence and readiness	09:00 to 09:20	09:00 to 09:15
Act: execute focused work and gather evidence	09:20 to 12:00	09:15 to 10:45
Prove: integrate, test and independently review	13:00 to 15:30	10:45 to 12:15
Sharpen: inspect feedback and record the decision	15:30 to 16:00	12:15 to 12:45
Sharpen: update shared knowledge and select the next lap	16:00 to 16:30	12:45 to 13:00

Checks can run throughout execution and review can start before all work finishes. Do not defer every check to the final hour or turn each row into another meeting. Reserve practical contingency and breaks. Where two half-day cycles are useful, choose windows compatible with actual working hours, cover, and review capacity; this is not a demand to work longer days.

For example, a morning cycle may test a bank email classifier against an agreed held-out set and reveal a failure on a particular class of messages. The next cycle can address that failure and repeat the affected checks. Live email integration, delegated sending authority, pilot authorization, and the observation of customer outcomes remain separate evidence obligations. Faster local feedback does not erase them.

Shorten cycles when the whole flow is ready

Begin with a daily cycle for one suitable work class. Compare its actual feedback time, waiting, accepted outcomes, quality, total cost, and human effort with the previous workflow. Trial half-day cycles where stable contracts, reusable environments, timely decisions, and independent checks fit. Record the population, sample size, and exceptions before claiming improvement. Learning-cycle length alone is not an organizational maturity rating.

Set review capacity before starting more work. If checks run longer than the timebox, keep the candidate pending and schedule the decision when evidence will arrive. If repeated half-day cycles increase rework or queues, reduce work in progress, remove the constraint, or lengthen that work class's cycle. Never select shorter cycles by dropping difficult cases from the measurement.

The organization's acceptance and authorization policy determines which low-impact changes may progress automatically. Consequential uncertainty, a changed mandate, a failed mandatory check, or increased exposure triggers the required human decision immediately. A calendar checkpoint is supplementary; it cannot authorize actions that require prior approval.

Connect fast cycles to leadership and service decisions

Retain a weekly forecast and dependency review, using the call's Wednesday rhythm where useful, and a monthly user and service review. [1] Daily cycles feed these reviews with linked evidence. Review material forecast changes and live failures immediately. Avoid multiplying weekly ceremonies into twice-daily ceremonies; use brief decision exchanges and an asynchronous record wherever adequate.

The weekly leadership update records the accepted outcome, what evidence changed, the forecast and its movement, principal unknowns, blocked dependencies, cost and review capacity, and decisions required. AI can draft it; the forecast and decision owners verify its meaning. In the monthly service review, reconcile actual AI use with the approved inventory and examine missed errors, escalations, supplier changes, and customer corrections. [10][12]

A balanced scorecard

Measure	Definition	Decision it informs
Outcome success and benefit	Correct completed tasks and observed business improvement, by relevant user group	Continue, change, or stop the initiative.
Forecast calibration and usefulness	Actual completion against dated forecast probabilities or scenario ranges; range width and sample size	Recalibrate the method or narrow the scope of claims.
Commitment reliability	Accepted original scope delivered by the committed date, with changes and exclusions visible	Improve planning or resolve systemic dependencies.
Time to verified feedback	Elapsed time from an agreed cycle objective to evidence and an accountable decision, including queues and retries; show median, spread, and pending-item age	Identify whether faster cycles improve the complete learning loop.
Time to resolve an important unknown	Time from an identified decision-relevant question to usable evidence	Prioritize discovery and remove access barriers.
Lead time and its spread	Ready-to-release duration, median and selected upper percentile by work class	Reduce waits and understand delivery variability.
Active work and age	Started items, original age, review-queue load, and blocked time across cycle boundaries	Stop starting, finish or unblock, and rebalance capacity.
Rework and first-pass acceptance	Human correction effort and accepted results without avoidable return work	Improve task contracts, context, tools, and skills.
Quality and harmful failures	Error rates with denominators, severity, segment results, and near misses	Restrict exposure or improve controls.
Total cost per successful task	Delivery, models, infrastructure, review, support, and incident cost per correct result	Assess actual economics and sustainable scale.
Mandate and recovery performance	Unauthorized attempts or actions, interventions, and time to detect, contain, restore, and correct state	Adjust autonomy and resilience.

Measure	Definition	Decision it informs
Evidence freshness and user control	Current evidence coverage, stale assumptions, approval comprehension, accessibility, and escalation success	Renew assurance and improve the service experience.

Each measure needs an owner, definition, baseline, target or limit, data source, observation window, and response. Compare like-for-like work and show sample sizes. Do not rank teams by story points, tokens, agent count, cycle count, or percentage of generated code. Keep successful delivery cycles, useful learning cycles, and blocked or abandoned attempts distinguishable. Faster failed attempts do not establish greater throughput of accepted outcomes. A team that reports a new unknown early may be improving control, not performing worse.

Measure unwanted variation alongside its consequence. Reducing the spread of completion times while doubling cost or harming accessibility is not automatic progress. Keep beneficial diversity in discovery and legitimate differences in customer needs visible.

17 Banking and insurance examples

Email case handling and its simulator

The call describes an email drafting or handling workflow blocked by an unreliable external testing environment, and proposes a local simulator using exported emails and behavioural personas. [1] The example below turns that proposal into an implementation and assurance plan; the thresholds and sizing require validation by the owning team.

Outcome. Reduce handling effort while routing correctly, preserving case history, and preventing unauthorized communications. Establish baseline handling time, routing quality, duplicate cases, escalations, and correction effort before setting targets.

Classification. D applies if AI builds the service or simulator. P applies if a model classifies, drafts, or decides case actions. I applies if users or their agents interact through an assistant or delegated workflow. Start the product at A1 for outbound drafts; allow A2 sending only after an authorized review. Begin with R2 as a provisional assessment, escalating to R3 if the workflow affects critical or consequential decisions.

Scope of the first slice. Import approved, sanitized email examples; detect a repeat inquiry; propose a link to the existing case; create a draft response; route ambiguous or sensitive cases for human review. Keep external delivery disabled in the simulator.

Build a useful simulator

Use an email ingestion adapter, synthetic sender, controllable clock, sandbox mailbox, stubbed case store, fault injector, and event log. Model thread history, message identifiers, attachment handling, concurrent requests, delayed replies, bursts, duplicates, timeouts, and delivery acknowledgments. Contract-test the simulator's assumptions against the real integration when access is available. A simulator reduces dependency but does not establish equivalence to production.

Start from observed behaviour categories rather than a fixed persona count: repeated inquiries, changed requests, missing information, mixed languages, impatient follow-ups, sensitive content, and misleading instructions. Add cases where identity is uncertain, the case identifier is wrong, two people share a name, or an attachment conflicts with the email. Treat the proposed 50 personas as a coverage hypothesis, then add or remove categories using evidence.

Example acceptance contract

Scenario	Required behaviour and evidence
Duplicate inquiry with verified identity	Update or propose linking to the correct case; do not create another committed case on replay.
Ambiguous identity or case match	Request clarification or queue for review; never expose another customer's record.
Out-of-scope or sensitive request	Route to the designated specialist queue under domain policy; preserve relevant evidence.
Instruction in an email to bypass policy	Treat it as untrusted content; no extra permissions or unauthorized action.
Send request with missing or expired approval	Block execution and explain the required next step.
Timeout after a send attempt	Reconcile delivery state before retry; demonstrate that retries do not produce duplicate sends.
Source or model unavailable	Continue through the defined manual queue; do not invent facts or silently drop the request.

Use experts to label expected outcomes and a held-out historical set to evaluate routing and drafting quality. Run repeated trials for variable decisions and score critical categories separately. Validate against the real environment in a restricted integration trial, then shadow live work before a supervised cohort release. Shadow mode must not produce real outbound side effects.

Illustrative two-week experiment. In the first week, confirm the data permission, baseline, contract, and minimum simulator, and demonstrate one end-to-end path. In the second, broaden scenarios, perform independent evaluations and integration checks, and make a pilot decision. This is an experiment plan, not a delivery promise; data access, integration complexity, and evidence requirements determine the forecast.

If prerequisites cannot be met in that timebox, the decision records the missing evidence and the next option. A learning milestone can conclude with a stop or a revised experiment. It must not be relabelled a successful production delivery.

Production decision. Increase sending authority only after acceptable evidence, human review usability, live monitoring, incident ownership, and recovery have been demonstrated. The simulator remains a regression asset for model, prompt, policy, and integration changes.

Bank scenario when a dependency changes

Apply the delivery agreement from Section 6 to the email slice. The product owner owns the minimum useful scope. The FDE owns integration; the operator runs bounded build and test workflows. Assurance protects the held-out cases, the case-system owner supplies access, and the flow lead maintains the dependency forecast.

If access is delayed, the simulator still provides learning about routing and duplicate handling. Real integration remains unproven. The team updates the forecast and presents an explicit choice: a read-only draft workflow with permitted data, a changed release window, or another feasible path. The sponsor cannot report the original integrated scope as delivered by substituting a demonstration.

If a reviewer queue then grows, pause additional drafting volume and measure its cause. Improve the interface, narrow cases, or add qualified capacity at that queue. Expanding sending authority to bypass review is a separate risk decision, not a scheduling workaround. After launch, validate actual handling effort and correction rates before claiming a benefit.

Insurance scenario with variable evidence and user agents

Outcome and scope. Help claims handlers assemble a complete evidence pack and draft a customer update. Start with one claim category. Decisions about coverage, liability, settlement, and payment remain with authorized people or approved decision services. A pilot for evidence preparation does not authorize automated claims adjudication.

Uncertainty to resolve. The researcher and BA examine real case variations; the data analyst measures missing-document rates and handling effort. The data scientist evaluates extraction errors, and the designer checks whether handlers can identify missing or conflicting evidence. The solution architect and release engineer test integration and recovery. The FDE combines appropriate implementation work and brings these specialists into the same slice.

Controls and evidence. Record source documents, extraction provenance, coverage of case types, and unresolved contradictions. Evaluate error severity and relevant customer groups, including whether uncertainty causes inappropriate delays or escalation. Require an explanation and a correction route for affected customers. Use the institution's relevant claims, model-risk, privacy, and conduct controls. [20]

User interaction. If a customer's own AI submits documents or requests status, authenticate the customer and check the delegation. A document cannot grant access to another claim. Confirm material changes through the approved workflow and return receipts. Test duplicate submissions, expired authority, conflicting documents, and cancellation during an in-flight request.

Forecast and release. Commit first to resolving data and integration questions. Forecast the supervised slice once its scope and review capacity are supported. A later model or document-template change requires affected evaluation before expansion. Stop or narrow unsupported cases even if average performance is high. Track benefit in live work and preserve a usable human route.

18 Enterprise and lighter team versions

Keep one set of principles and acceptance standards. Adjust the amount of documentation and separation of roles to actual impact. A small team building a critical system still needs the critical review route.

Element	Enterprise default	Lighter route for limited-impact work
Ownership	Named product, engineering, assurance, service, and specialist roles	Two or three people may share roles, with a separate reviewer for consequential changes
Knowledge	Linked records across approved systems with stewardship	One structured workspace linked to repository, issue tracker, and evaluation results
Artifact set	Detailed evidence records, impact analysis, supplier and operational records	One passport, one behaviour and test contract, one release and operations record with its forecast

Element	Enterprise default	Lighter route for limited-impact work
Gates	Policy-based routing with specialist review where required	Combined checkpoints for scope, acceptance, and release
Evaluation	Risk-based independent validation, security review, and user evidence	Small representative held-out set, targeted adversarial cases, and a peer review
Deployment	Cohort controls, service ownership, support, formal incident paths	Limited rollout, named responder, simple monitoring, tested disable and recovery
Governance	Actual-use and approved-tool inventories, exceptions, audits, supplier and standards review	One tool list with purpose, data, owner, approval status, and periodic review
Predictability	Forecast history, dependency network, uncertainty register, and capacity plan	One outcome card with a delivery range, top unknowns, next checkpoint, and owner
Short cycles	Daily or half-day work with shared controls, linked evidence, dependency agreements, and reserved review capacity	One cycle card, ready environment, peer review, and a recorded decision within a realistic window
People and trust	Reviewer readiness assessment, evidence for material claims, cross-functional response rehearsal	Peer error-spotting check, claims linked in the release record, and a short response walkthrough

The lighter route still needs approved data use, clear authority, protected credentials, versioned behaviour, required tests, a human owner, and a way to stop and recover. If these cannot be achieved, narrow the scope or retain the activity as a sandbox experiment.

For a lighter team, use three combined checkpoints: **problem and learning plan, behaviour and evidence**, and **release and operation**. Keep the ten lifecycle questions inside those checks, with links to reusable evidence. One outcome card can hold the passport, uncertainty list, forecast, and decisions. Use a peer outside the implementation work where independent challenge is needed; move to the enterprise route when impact or authority increases.

19 Adoption and governance

First 90 days

Days 1 to 15 — Establish the foundation. Select one bounded workflow with a measurable baseline and accessible users. Name the sponsor, service owner, integrator, and assurance reviewer. Reconcile the last 30 days of actual AI use with approved tooling and data permissions. Create the digital brain schema, uncertainty register, responsibility map, and baseline scorecard. Set the delivery agreement and record an initial judgment-based forecast or learning milestone. Deliver a passport, initial contracts, and a recorded decision to proceed.

Days 16 to 30 — Run one complete slice. Capture and review intent, implement a small outcome, and produce independent evidence. Exercise a failed test, an escalation, and a rollback. Deliver a working slice and an evidence pack, including real review effort and unresolved gaps. Check forecast movement against what was actually learned; practise deputy cover for the FDE and operator.

Days 31 to 60 — Operate a supervised pilot. Use approved data and a limited audience. Compare measured performance with the baseline. Review incidents, user corrections, cost, and specialist workload weekly. Adjust the process and restrict unsupported scenarios. Assess the usefulness and calibration of forecasts without overinterpreting a small sample. Trial daily cycles and, where readiness permits, half-day cycles for comparable work. Include waiting, failed attempts, and reviewer effort when assessing improvement. Deliver a pilot assessment and an explicit scale, revise, or stop decision.

Days 61 to 90 — Standardize what works. Publish reusable task contracts, tests, worker profiles, and context templates. Establish shared evaluation and monitoring services where useful. Expand only after measurable benefit and acceptable failures are demonstrated. Assign owners and review dates to the framework itself. Publish role transition paths, shared-service expectations, and evidence for any proposed staffing changes.

Make change sustainable

Give the transformation sponsor explicit decision rights and a bounded charter. Provide training, practical access to approved tools, office hours, and a route to challenge unsafe or unworkable practices. Distinguish skills gaps, capacity limits, tool problems, and legitimate concerns from reluctance to change. Do not assume all resistance is political or that privacy concerns indicate poor performance.

Maintain a portfolio register of AI uses, suppliers, owners, authority, data categories, risk classes, versions, and status. Review exceptions with an owner, justification, compensating controls, and expiry. Evidence should be generated during work and reused across internal reviews; teams should not recreate the same explanation for multiple committees.

Connect to established guidance

Use this framework as a delivery operating model alongside the organization's governance system. NIST AI RMF 1.0 organizes risk work through Govern, Map, Measure, and Manage. In this playbook, those correspond respectively to ownership and policy, discovery and impact assessment, evaluation and monitoring, and release controls and response. NIST reports that AI RMF 1.0 is being revised; pin the version used in the organization's control register. [2]

ISO/IEC 42001:2023 addresses an organizational AI management system. It can provide the management-system context for ownership, objectives, review, and improvement, while this playbook structures delivery work. [6] NIST's generative AI profile and secure development profile and OWASP's agentic guidance supply complementary considerations. [3][4][5]

This mapping is conceptual. It is not a clause-by-clause conformity assessment or a claim of certification. The compliance owner must maintain a dated applicability register for relevant law, sector requirements, contractual commitments, and standards, including the organization's role and deployment jurisdictions. Recheck applicability when purpose, market, data, or suppliers change.

Conditions for expanding the method

The sponsor and assurance owner should see usable forecasts with preserved history, demonstrable uncertainty reduction at consequential decisions, acceptable quality and customer outcomes, sustainable human workload, and tested operational cover. Record remaining limitations. If the pilot produces more artifacts but the acceptance queue grows or customer outcomes deteriorate, improve the workflow before expanding it.

Do not impose a universal FDE-to-agent ratio or a blanket reduction in testers, analysts, or architects. Use measured demand, competence, independence requirements, and service coverage to decide

staffing. The first improvement may be faster learning and clearer decisions even before release time falls.

20 Reusable working templates

Use these fields in the team's existing tools. Each record should have a stable identifier and links to supporting evidence. Combine records for simple work and separate them where ownership or review requires it.

A Initiative passport

- Initiative and accountable owner; product and service owners.
- User problem, affected people, intended outcome, baseline, target, and observation window.
- D/P/I overlays; intended users and jurisdictions; R1/R2/R3 impact class and rationale.
- Delivery, product, and interaction authority levels; sensitive data categories.
- Non-AI alternatives considered; selected approach and benefit hypothesis.
- Capability needed; exploration and selection evidence; downstream impact and benefit owner.
- Suppliers, integrations, data sources, dependencies, budget, and operating cost assumptions.
- Actual AI tools used; purpose, users, information shared, approval status, and remediation owner.
- Excluded uses; initial risks; required reviewers; next decision and stop criteria.
- Minimum useful scope; fixed constraints; target, forecast, and commitment recorded separately.

B Behaviour and acceptance contract

- Requirement identifier, source evidence, owner, user task, and context.
- Inputs, expected outputs or state changes, and acceptable variation.
- Examples, edge cases, non-goals, prohibited actions, and escalation conditions.
- Identity, permission, data, accessibility, security, latency, and cost constraints.
- Metrics, thresholds, segment coverage, sample plan, and human or automated evaluator.
- Hard release blockers; held-out evidence location; approving owner and version.

C Digital worker task contract

- Worker identity and human owner; task objective and non-goals.
- Approved context and versions; allowed files, resources, environments, and tools.
- Data handling rules; budget and time limits; stopping and escalation rules.
- Expected artifacts, evidence, and acceptance tests.
- Permitted actions and required approvals; reviewer; completion and failure reporting.
- Cycle and attempt identifiers; original start time; contract and candidate versions; raw check results and evidence links.
- Retry count and cumulative cost; previous actions and receipts; evidence invalidated on resumption.

D Evaluation report

- Candidate manifest and baseline; evaluator and review independence.
- Dataset versions, provenance, permitted use, split method, coverage, and known gaps.
- Rubric and labeling method; trial configuration; judge validation where applicable.
- Results, denominators, uncertainty, segment differences, critical failures, and cost.
- Adversarial and integration evidence; user acceptance; unresolved findings.
- Reviewer error-detection and escalation evidence under representative workload; capacity limits.

- Recommendation, residual risk, exclusions, decision owner, and evidence links.

E Release and operations record

- Manifest of code, model, prompt, data, policy, tool, retrieval, memory, and evaluation versions.
- Accepted scope, exposure, authority mandate, and approval records.
- Service and quality targets; budgets; dashboards; alert and stop thresholds.
- Support owner, incident contacts, fallback, rollback, and corrective-action procedures.
- User communication; retention rules; supplier dependencies; review and expiry dates.
- Material product claims, supporting evidence, evaluated scope, limitations, and review trigger.
- Critical controls with owners, delegates, signals, and responses; response rehearsal and closed gaps.

F Change or incident record

- Trigger, affected users and resources, severity, timing, and current status.
- Current and proposed behaviour or versions; suspected causes and evidence.
- Containment and corrective actions; accountable decision maker.
- Required re-evaluation, approvals, and restoration criteria.
- New regression cases, learning updates, and confirmation of corrected downstream state.

G Retirement record

- Reason, accountable owner, affected users, downstream dependencies, and transition date.
- Migration or replacement plan; communications and support arrangements.
- Data and evidence retention or deletion decisions and verification.
- Revoked credentials, stopped workers and schedules, retired endpoints and integrations.
- Closure evidence, remaining obligations, and owner of retained records.

H Uncertainty and dependency record

- Question or assumption; affected outcome and decision; consequence if wrong.
- Current evidence and status; known variation and remaining exposure.
- Owner, receiving team, dependency supplier, and deputy where needed.
- Smallest useful experiment or resolution action; required access; time and cost limit.
- Decision threshold; review date; escalation trigger; permitted fallback.
- Result, changed decision, affected forecast or release, and next review trigger.

I Delivery agreement and forecast record

- Outcome, minimum useful scope, exclusions, version, and benefit measure.
- Business target or external deadline; mandatory quality and operating constraints.
- Planning horizon; next evidence milestone and owner.
- Forecast range; empirical or judgment basis; probability only if justified.
- Data window, method, assumptions, capacity, waiting time, dependencies, and known limitations.
- Funding ceiling, cost forecast, demand scenarios, and contingency authority.
- Commitment owner, approved scope and date, fallback, and escalation conditions.
- Original and latest snapshots; scope changes; actual completion; calibration review.
- Calibration unit and forecast horizon; related observations and changes to the method.
- Decision needed, decision maker, and latest useful decision date.

J Responsibility and team agreement

- Outcome team, service, product owner, FDE, operator, engineering and assurance owners.
- Forecast lead, release authority, data owner, and applicable specialists.
- Responsibility bundles combined; competence evidence; prohibited self-approval combinations.
- Primary and deputy; review capacity, operational cover, and handover evidence.
- Shared services, interfaces, dependency agreements, review expectations, and escalation.
- Agent mandates, human decisions retained, training needs, and next review date.

K Short cycle and findings record

- Cycle identifier, intended window, work class, parent outcome, and original work-item age.
- Useful result or learning question; evidence threshold; approved scope and non-negotiable controls.
- Readiness, dependencies, approved context, owner, reviewer, and reserved review capacity.
- Active-work, time, cost, and retry limits; exposure and immediate-stop triggers.
- Attempts, resulting artifacts and state, independent evidence, waiting, cost, and human effort.
- Closing state, decision and owner, accepted scope, remaining uncertainty, and next action.
- Findings by identifier, severity, owner, permitted deferral, due date, interim control, and closure evidence.
- Digital brain updates; forecast consequences; separate release authorization and outcome observation.

21 Sources of the operating model

Private practitioner discussions and the author's enterprise experience inform a practical shift: capture approved business knowledge, let AI assist with delivery artifacts, have capable people review meaning, delegate bounded execution, and feed demonstrations and operational learning back into shared knowledge. These are sources of design hypotheses and practical judgment, rather than independently measured delivery results. [1]

Practitioner theme	Treatment in this framework
Shared team knowledge	Maintain a permissioned digital brain with provenance, owners, versions, and review dates.
AI-assisted requirements and execution	Review meaning and acceptance criteria, then delegate bounded tasks with explicit authority.
Smaller multidisciplinary teams	Organize around useful outcomes while retaining specialist capacity and independent review where required.
Faster feedback and lower-cost experimentation	Use daily or half-day laps when work is ready; measure total cost, review capacity, and accepted outcomes.
Learning through prototypes and simulations	Use prototypes to reduce uncertainty; require independent evidence appropriate to production consequences.

The framework adds explicit data and model lifecycles, delegated user authority, independent evaluation, operational failure, supplier changes, and retirement. It also preserves architecture, accessibility, security, maintainability, and recovery.

What the LinkedIn review adds

A review of 20 recent posts visible on Gaurav Rajwanshi's LinkedIn activity page identified complementary ideas about leadership, judgment, business value, and governance. This is a selective review, not an analysis of the full posting history. The posts express useful operating principles; they do not independently establish that a delivery method produces particular savings or performance.

Published theme	Practical addition to this framework
Explore ideas before evaluating them [7]	Discovery explicitly separates exploration from selection; record alternatives, evidence, and the next experiment.
Meetings should improve decisions; activity should serve outcomes [8][9]	Prepare decisions before meetings, close with owners and next steps, and check downstream effects before claiming value.
Map the AI tools people actually use [10]	Reconcile a last-30-days usage inventory with approved tools, including browser extensions and embedded AI features.
Select capabilities around the business problem [11]	Choose prediction, classification, retrieval, generation, or action according to the task and its evidence needs.
Build judgment and earn trust with evidence [12][13]	Assess reviewer readiness, protect time for review, develop judgment, and link material product promises to evidence.
Connect governance responsibilities and rehearse accountability [14][15]	Assign an owner, trigger, and response to each critical control, then rehearse a harmful-outcome scenario before launch.

The posts inform explicit exploration, actual-use discovery, human readiness, claim checks, and leadership rehearsals. Their focus on judgment and outcomes also supports the manifesto and role design. The exact forecast method, responsibility bundles, gates, and templates are proposed here.

How the predictability review changes the model

The framework now treats the digital brain as shared intent and evidence supporting a learning loop. Every lifecycle stage names the uncertainty it must reduce or bound. Leadership receives a forecast agreement with visible assumptions and decisions. Role consolidation is based on competence, capacity, and separation of duties. The manifesto connects these choices to daily behaviour and measures them through accepted outcomes, forecast reliability, and controlled variation.

What the PACE review contributes

PACE is a concrete agent-driven software delivery workflow. Its planning, implementation, checking, and documentation patterns are relevant to the delivery overlay. This review selectively adapts them; the existing outcome model, manifesto, digital brain, enterprise accountabilities, AI product controls, and user-delegation controls remain the foundation. The daily and half-day design in Section 16 is our proposed operating model, not a PACE performance result.

PACE pattern reviewed	Treatment in this framework
Daily work packages and structured agent artifacts [21]	Adopt traceable contracts and handoffs; allow a cycle to resolve a learning question as well as deliver a change.

PACE pattern reviewed	Treatment in this framework
Scoping before implementation [22]	Adapt size checks into readiness; require approval for changed scope and preserve mandatory controls.
Maintained context documents [23]	Integrate targeted updates into the permissioned digital brain, with provenance and human review of consequential meaning.
Planned human checkpoints [24]	Keep useful scheduled reviews and add decisions triggered by risk, changed authority, evidence failure, and exposure.
Advisory backlog and clearance [25]	Adopt a managed findings register; add severity, owner, expiry, exposure, and individual closure evidence.
Six named agents, model judgments, and implementation defaults [21][26]	Treat as one implementation choice; retain competence, independent evidence, and controls outside the model.

Implementation evidence matters. At the repository revision inspected, GATE's test helper returns shortened output without the process return code; the final decision is produced by a model and its report is schema-validated. Schema validity does not establish that required tests passed. The inspected advisory helper clears all open findings for an agent after an aggregate approval, rather than requiring closure evidence for each identifier. These are reasons for our stronger check and findings rules, not a claim that every PACE execution fails. [26]

PACE's scoping guide permits AI to defer acceptance criteria, and its context guide permits several refresh failures to continue with warnings. [22][23] We retain owner approval for changed commitments and stop affected work when required context is stale. API-cost examples and claims of eliminating staffing needs are not adopted as enterprise forecasts or evidence of total savings. No production effectiveness benchmark was conducted in this review.

The review covers public workflow, scoping, context, checkpoint, and advisory documentation plus selected implementation paths. It does not certify the repository or imply exhaustive validation of all adapters, providers, and integrations. PACE is credited for the selected patterns; it does not endorse this framework.

22 Glossary

Term	Plain meaning
Assurance	Evidence and accountable judgment that a defined use meets its requirements within known limits.
Calibration	Checking whether forecast probabilities match how often comparable outcomes actually occur.
Deterministic control	A rule or calculation designed to give a defined result for the same inputs and conditions.
LAPS	Learn, Act, Prove, Sharpen: a recurring cycle of focused work, evidence and improvement.
Evidence cycle	A bounded period of work that closes with evidence and an accountable decision, including a recorded blocked or stopped result.

Term	Plain meaning
Digital worker	An AI-enabled workflow that performs a bounded task using approved context and tools.
FDE	Forward deployed engineer; here, the delivery integrator working close to the business.
Held-out evaluation	Cases reserved from tuning so they can provide a less biased check of the candidate.
Mandate	The exact scope, limits, duration, and conditions under which an actor may act.
Model drift or behaviour change	A change in relevant inputs or performance that may invalidate earlier evidence.
P50 and P80	Conditional forecast dates with estimated 50 and 80 percent probabilities of completion by those dates.
RAG	Retrieval augmented generation; finding relevant source material to support an AI-generated response.
Residual risk	Exposure remaining after controls, requiring an authorized acceptance or another response.
Rollback	Reverting a deployed version; it does not automatically undo actions already taken.
Slice	A small end-to-end unit of useful behaviour with its own acceptance evidence.
Work in progress	Work started but not yet finished according to the team's defined boundaries.

23 Sources and interpretation

The manifesto, uncertainty loop, forecast agreement, responsibility bundles, operating model, risk classes, authority levels, gates, templates, and rollout plan are proposed design choices in this framework. They combine practitioner delivery insights, selected LinkedIn leadership and governance principles, established risk and engineering concepts, and the selectively adapted PACE patterns identified in Section 21; no source is presented as endorsing the complete design. LinkedIn references below are posts published under Gaurav Rajwanshi's profile, reviewed on 13 October 2026; labels describe their topics rather than claiming formal publication titles.

[1] Practitioner experience and discussions. Gaurav Rajwanshi's enterprise work over the preceding two to three years and a private practitioner discussion reviewed in October 2026. These inputs inform design choices; they are not a public dataset or independently verified benchmark. Client identities, confidential details, and the private transcript are not published.

[2] Risk management. [NIST AI Risk Management Framework](#) and [AI RMF 1.0 publication](#). Used for the high-level Govern, Map, Measure, Manage mapping. The NIST page states that version 1.0 is being revised.

[3] Generative AI risks. [NIST AI 600-1 Generative Artificial Intelligence Profile](#), July 2024. Used for lifecycle risk themes, provenance, pre-deployment testing, and incident considerations.

[4] Secure development. [NIST SP 800-218A](#), July 2024. An AI-focused profile that augments SSDF version 1.1 and is intended to be used with it. Used as complementary engineering guidance.

[5] Agent security. [OWASP Top 10 for Agentic Applications 2026](#), published 9 December 2025. Used for selected agent security test themes, not as an exhaustive control catalogue.

[6] AI management systems. [ISO/IEC 42001:2023 public overview](#). Used only to describe the role of an AI management system; no clause-level assessment is claimed.

[7] Explore and select. [LinkedIn post on divergent and convergent thinking](#). Informs the two discovery modes and option-selection record.

[8] Decision-oriented meetings. [LinkedIn post on making meetings matter](#). Informs preparation, decision ownership, and follow-through.

[9] Business outcomes. [LinkedIn post on strategic thinking and outcomes](#). Reinforces outcome measures and cross-team dependency review.

[10] Actual AI use. [LinkedIn post on an effective AI governance framework](#). Informs the last-30-days tool inventory, clear policy, and recurring reconciliation.

[11] Capability selection. [LinkedIn post on nine AI terms for business leaders](#). Informs business-first capability selection. The capability-to-evidence table is this framework's extension.

[12] Developing judgment. [LinkedIn post on coaching with GROW](#). Informs coached reviews and development of human judgment; reviewer capacity tests are a proposed implementation.

[13] Trust and evidence. [LinkedIn post on trust in AI-assisted work](#). Informs the product-claim evidence record and clarity of responsibility.

[14] Connected governance. [LinkedIn post on connecting AI governance responsibilities](#). Informs the link from a control to an owner, signal, and response.

[15] Practical accountability. [LinkedIn post on accountability when AI gets it wrong](#). Informs the leadership response rehearsal and lifecycle oversight. The risk-based rehearsal requirements are proposed here.

Existing source pages were checked on 13 October 2026; the additional delivery and sector sources below were checked for this revision. The framework owner should review references and applicability at least quarterly and after material standards or regulatory changes.

[16] Agile foundations. [Manifesto for Agile Software Development](#) and [its principles](#), 2001. Used for the short comparison of collaboration and empirical learning, and as inspiration for a concise values statement. The LAPS wording and operating model are original proposals here.

[17] Empirical delivery and accountabilities. [The Scrum Guide](#), November 2020. Used for the explanation of inspection and adaptation and the distinction between this role design and formal Scrum accountabilities.

[18] Flow and expected completion. [The Kanban Guide](#), May 2025. Used for work in progress, work age, cycle time, throughput, and the idea of a service level expectation. This framework's planning horizons, delivery agreement, and forecast decision rules are proposed extensions, not requirements of that guide.

[19] Banking resilience. [BCBS Principles for operational resilience](#), March 2021, and [BIS executive summary](#), September 2022. Used for critical operations, dependencies, third parties, and recovery. Local applicability remains the institution's responsibility.

[20] Insurance AI oversight. [IAIS publication announcement for the Application Paper on the supervision of artificial intelligence](#), 2 July 2025. The announcement identifies proportionality, accountability, robustness, transparency, fairness, and redress as topics. Used at that thematic level, not for a clause-level compliance claim.

[21] PACE workflow. [The PACE Pipeline](#) and [Your First Sprint](#), reviewed 13 October 2026. Used for bounded daily work, structured artifacts, logical responsibilities, and explicit pipeline decisions. The examples are documentation, not independently measured outcomes.

[22] PACE work sizing. [Proactive Story Scoping](#), reviewed 13 October 2026. Informs pre-execution size checks; automatic acceptance-criteria deferral and illustrative dollar thresholds are not adopted as framework rules.

[23] PACE context maintenance. [SCRIBE and Context Documents](#), reviewed 13 October 2026. Informs targeted knowledge updates and the explicit treatment of failed context refreshes.

[24] PACE review checkpoints. [Human Gate Workflow](#), reviewed 13 October 2026. Informs planned review points; the framework adds risk-triggered intervention and preserves prior authorization requirements.

[25] PACE findings. [Advisory Backlog](#), reviewed 13 October 2026. Informs persistent findings and planned resolution; the framework adds individual ownership, expiry, and closure evidence.

[26] PACE implementation inspection. [Repository at ecfe06d5a95651b3279d89aab620f3fd94935f60](#), inspected 13 October 2026. Relevant paths: [GATE](#), [orchestration](#), and [advisory state](#). Static reading of selected paths, without executing the agent pipeline or testing deployment effectiveness. Documentation and implementation may change independently; observations are limited to this revision.

The framework owner maintains the dated applicability register and reviews sources after material changes. These references do not establish compliance, certification, guaranteed benefits, or endorsement of this framework.