ClearWork Transformation Playbook

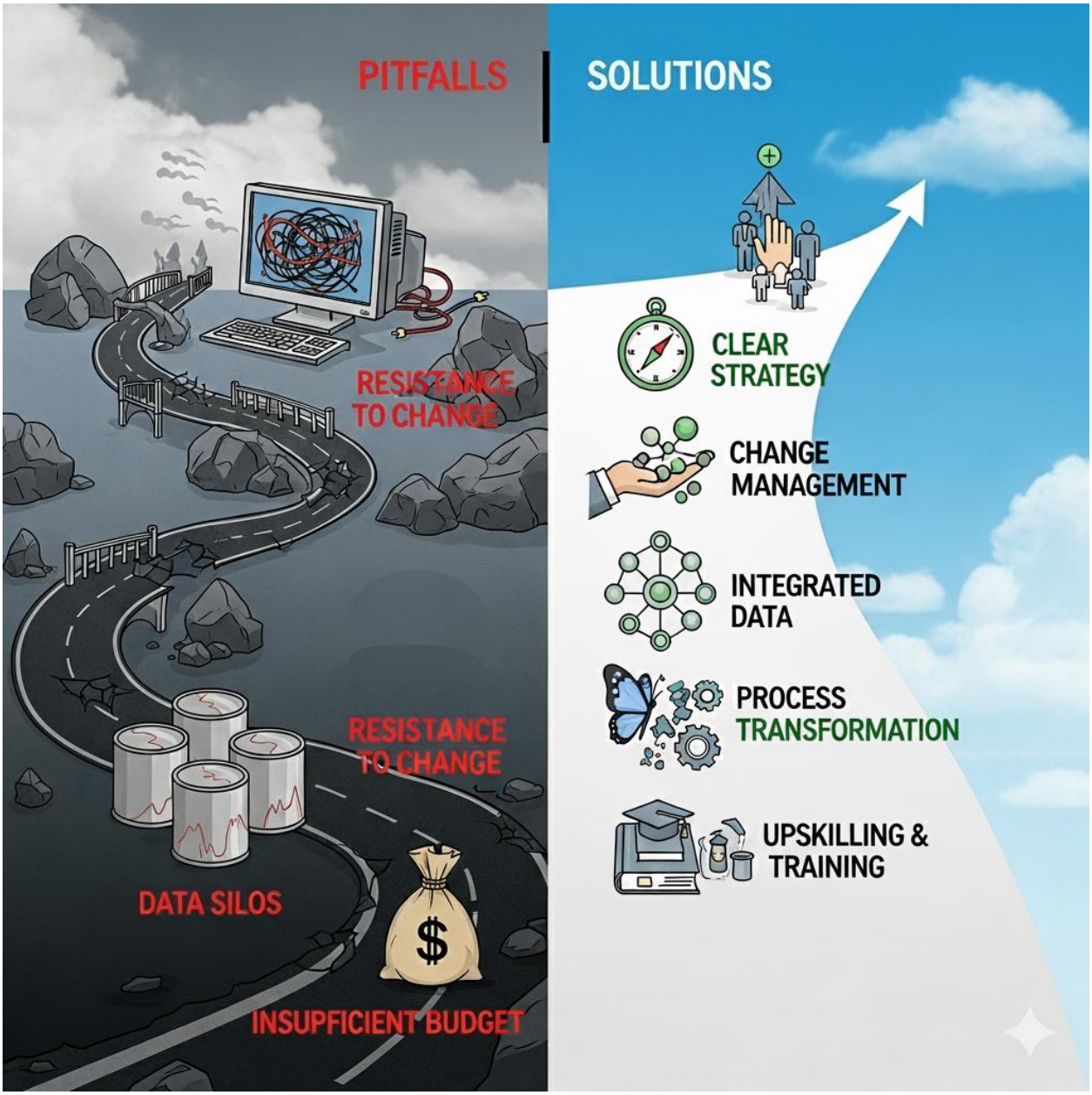
A practical, end-to-end guide to plan, execute, and sustain successful digital transformation—grounded in process intelligence, automatic process mapping, task mining, and responsible AI/agent adoption.

September 2025

0) How to Use This Playbook

- **Audience:** CIO/CTO, PMO/Transformation Office, Process Excellence leaders, Product/IT Owners, Compliance & Data Governance, frontline managers.
- **What you get:** A repeatable operating model, decision frameworks, and ready-to-use templates. Each section ends with **Action Steps** and **Artifacts**.
- **ClearWork fit:** Where to leverage ClearWork's process intelligence, business process mapping tools, and agent orchestration throughout.

1) Why Transformations Fail (and How to Avoid It)



Objectives: Establish a shared understanding of common failure modes and the leadership behaviors that flip the odds.

Key pitfalls:

- Tech-first, process-second (no clear business problem, weak value hypothesis).
- Vague decision rights; diffuse accountability; dashboard theater vs real outcomes.
- Underfunded **change management**; sponsor fatigue; fragmented communications.

- Data and integration debt; security/privacy left to the end; AI/automation without guardrails.
- Benefits not tracked; no single source of truth for value realization; program drift.

Action Steps:

1. Write the **Value Hypothesis** (problem, affected metrics, targeted lift, payback).
    1. Identify the core problem – Gather evidence from performance data, user feedback, or compliance findings. Be specific (e.g., "Order processing takes 14 days vs. 7 days industry benchmark").
    2. Link to strategic goals – Map the problem to business objectives (e.g., revenue growth, customer satisfaction, compliance).
    3. Define affected metrics – Select measurable indicators (cycle time, NPS, cost per transaction, error rates).
    4. Set the targeted lift – Quantify the intended improvement (e.g., 40% faster processing, 25% fewer errors).
    5. Estimate payback – Project ROI using a cost-benefit calculation (investment vs. savings or revenue impact).
    6. Document in one page – Write the hypothesis clearly: "By automating onboarding steps, we will reduce onboarding time from 14 days to 7 days, raising NPS from 40 to 65 and saving $500K annually, with payback in 12 months."
2. Stand up a small **Transformation Office (TO)** with mandate, cadence, and rules of engagement.
    1. Appoint a sponsor – Assign an executive sponsor(s) with authority (CIO, COO, or equivalent).
    2. Define the mandate – Write down what the TO owns: scope definition, roadmap, benefits tracking, risk management, communications.
    3. Staff the core team – Start lean:
        1. TO lead (program manager)
        2. Process analyst(s)
        3. Change/adoption lead
        4. Data/metrics lead
    4. Set the cadence – Weekly working sessions, biweekly status updates, monthly executive reviews.

5. Establish rules of engagement – Clarify how the TO interacts with project teams, IT, and business owners (e.g., escalation paths, reporting formats, RACI alignment).
6. Communicate its role – Announce TO purpose and authority to stakeholders so it's understood and respected.

3. Define decision rights with a **RACI** (Responsible, Accountable, Consulted, Informed).
    1. List key decisions/tasks – e.g., approve process design, sign off on budget, choose AI tool, approve go-live.
    2. Identify stakeholders – Include executives, functional leaders, IT, compliance, end-users.
    3. Assign roles:
        1. R (Responsible): The doer(s).
        2. A (Accountable): The single owner with final authority.
        3. C (Consulted): Experts or groups providing input.
        4. I (Informed): Those kept updated.
    4. Validate with stakeholders – Hold a short workshop to confirm and resolve overlaps.
    5. Publish the RACI – Store it in a shared, visible place (e.g., ClearWork, Confluence, Teams).
    6. Review quarterly – Update as the transformation scales.
4. Choose a **governance spine** (SteerCo, Architecture Board, CAB/CCB, Data/AI Governance Council).
    1. List governance needs – Strategy, architecture, security, change control, AI/data ethics.
    2. Select forums:
        1. SteerCo (Steering Committee): Executive priorities, funding, risk.
        2. Architecture Board: Standards, integration, scalability, security.
        3. CAB/CCB (Change Advisory/Control Board): Production change approvals.
        4. Data/AI Governance Council: Data privacy, model risk, AI safety.
    3. Define scope & cadence for each – e.g., SteerCo monthly, CAB weekly, ARB biweekly.

4. Appoint chairs & members – Assign accountable executives and cross-functional leads.
5. Write terms of reference (ToR): Clarify decision rights, inputs, outputs, SLAs for decisions.
6. Integrate into TO workflow – Ensure decisions flow logically between bodies (no duplication).

5. Pre-commit to **benefits realization** KPIs and a measurement plan.
    1. Select benefit categories – Efficiency, revenue uplift, customer experience, compliance, employee engagement.
    2. Define KPIs per category – Cycle time, NPS, churn, SLA compliance, employee survey scores.
    3. Baseline current values – Measure where you are today (using system data, surveys, or benchmarks).
    4. Set targets & timing – Quantify what improvement is expected and by when.
    5. Assign owners – Each KPI has a named person accountable for measurement.
    6. Choose data sources & methods – e.g., CRM reports, HRIS data, ClearWork dashboards, surveys.
    7. Set review cadence – Monthly benefit check-ins, quarterly executive reviews.
    8. Integrate with reporting tools – Automate dashboards where possible for transparency.

Artifacts: Value Hypothesis one-pager; TO charter; RACI template; governance map; benefits register.

---

2) Readiness & Preparation

Objectives: Align strategy, scope, and governance before touching systems.

Workstreams:

- **Strategy & Scope:** Link to objectives and key results; define constraints; set non-functional targets (privacy, reliability, cost to serve).
    - **Link to OKRs (top-down and bottom-up)**
        - Map each initiative to a single **Objective** and 2–4 **Key Results**.

- Ensure at least one KR is **outcome-based** (e.g., cycle time, NPS, cost-to-serve) rather than activity-based.
- Create a **traceability view**: Initiative → KR → Owner → Measurement source.

- **Define constraints up front**
  - **Time:** pilot length, latest go-live date, blackout windows.
  - **Budget:** capex/opex limits, vendor ceilings, service-hours guardrails.
  - **People:** max % allocation per team; backfill plan for critical roles.
  - **Change velocity:** limit concurrent changes per function to protect adoption.
- **Set non-functional targets (NFRs)**
  - **Privacy:** data classification, masking rules for task mining capture, purpose limitation.
  - **Reliability:** uptime SLOs, RTO/RPO, incident response times.
  - **Cost to serve:** per-transaction or per-case cost targets; infra cost budget.
  - **Security:** least-privilege, auditability, log retention, environment segregation.
- **Anti-patterns to avoid**
  - "**Do everything this quarter**" scope; success metrics defined after go-live; NFRs left implicit.
- **Acceptance criteria (DoD)**
  - Each initiative has **one OKR link**, **documented constraints**, and **NFRs with numeric targets**.

- **Operating Model & Governance:** Clarify SteerCo vs TO vs PMO; define escalation paths; change control (CAB/CCB); architecture guardrails.
  - Clarify roles: SteerCo vs TO vs PMO
    - SteerCo: strategy, funding, risk acceptance, cross-business tradeoffs.
    - Transformation Office (TO): value tracking, cadence, unblockers, benefits realization.
    - PMO (optional): portfolio health, milestones, RAID register hygiene.
  - Define escalation paths

- 3 tiers max (Squad → TO → SteerCo) with decision SLAs (e.g., 48h/5 business days).
- Escalation template: decision needed, options, risk, recommended path, time cost.

- Change control (CAB/CCB)
  - Classify changes (standard, normal, emergency) with approval routes.
  - Require backout plan + user impact note for every production change.
- Architecture guardrails
  - Publish reference patterns (workflow engine, iPaaS, eventing, identity).
  - Maintain tech radar (adopt/trial/hold) to avoid tool sprawl.
  - Pre-approved integration & data patterns (source of truth, API standards).
- Anti-patterns
  - Governance by inbox; every forum decides everything; CAB as a rubber stamp without rollback.
- Acceptance criteria
  - Documented RACI of bodies, decision SLA matrix, change policy, guardrails deck.

- **Stakeholders & Sponsorship:** Map sponsors, champions, skeptics; create sponsor roadmap; plan engagement moments.
  - Map sponsors, champions, skeptics
    - Build a stakeholder heatmap (influence × support).
    - Identify persona-level needs and likely objections (time, risk, data, compliance).
  - Create a sponsor roadmap
    - Quarterly moments that matter (kickoff, value reviews, town halls).
    - Sponsor talk tracks tied to OKRs and user benefits (not just system features).
    - Pre-book sponsor time for escalations and celebrations.
  - Plan engagement moments
    - Champions network with office hours and early access.

- Short, role-based demos showing “what changes for me”.
- Feedback loops: “report a snag” in-app + monthly sentiment pulse.

- Anti-patterns
  - One email blast to “announce change”; silent sponsors; skipping skeptics until go-live.
- Acceptance criteria
  - Named executive sponsor, champions per function, and a 90-day engagement calendar.

- **Benefits & Metrics:** Select outcome KPIs per function; define baselines; choose cadence; set leading indicators.
  - Select outcome KPIs per function
    - Finance: days payable outstanding, cost per invoice.
    - Sales: cycle time to quote, win rate.
    - Support: first-contact resolution, average handle time.
    - IT/Engineering: DORA metrics, change failure rate.
  - Define baselines
    - Use last 3–6 months of data; note seasonality and anomalies.
    - Document data source and owner for each KPI.
  - Choose cadence & reviews
    - Monthly benefits check-ins; quarterly SteerCo value reviews.
    - Maintain a benefits register with forecast vs. actuals.
  - Set leading indicators
    - Adoption proxies (e.g., % users following new SOP path; time-on-task improvement).
    - Quality proxies (rework rate, exception volume).
    - Early risk flags (capacity alerts, backlog growth).
  - Anti-patterns
    - Vanity metrics; activity counts as outcomes; dashboards without owners.
  - Acceptance criteria

- For each initiative: baseline, target, owner, data source, review cadence documented.

- **Data & Integration Readiness:** System inventory, APIs, identity/permissions, data quality, lineage.
  - System & data inventory
    - For each system: purpose, data domains, owners, environments, RPO/RTO, SLAs.
    - For each dataset: schema, sensitivity, retention, quality score, source of truth.
  - APIs & integration patterns
    - Standardize on authentication, rate limits, pagination, error contracts.
    - Prefer event-driven where feasible; document idempotency expectations.
  - Identity & permissions
    - Centralize on SSO; define least-privilege roles for automations and agents.
    - Create service accounts with rotation policies and auditable scopes.
  - Data quality management
    - Define DQ rules (completeness, validity, uniqueness, timeliness).
    - Set up DQ monitors and alert thresholds with owners and SLAs to fix.
  - Lineage & cataloging
    - Track lineage for critical metrics (where it comes from, transforms, consumers).
    - Maintain a lightweight data catalog with business definitions.
  - Anti-patterns
    - Building integrations without identity model; hidden spreadsheets as source of truth; “we’ll fix data later.”
  - Acceptance criteria
    - Inventory completed, API standards agreed, RBAC defined, DQ rules live, lineage for top metrics.
- **Risk & Compliance:** Threat model; regulatory constraints (PII, retention, auditability); AI risk boundaries.

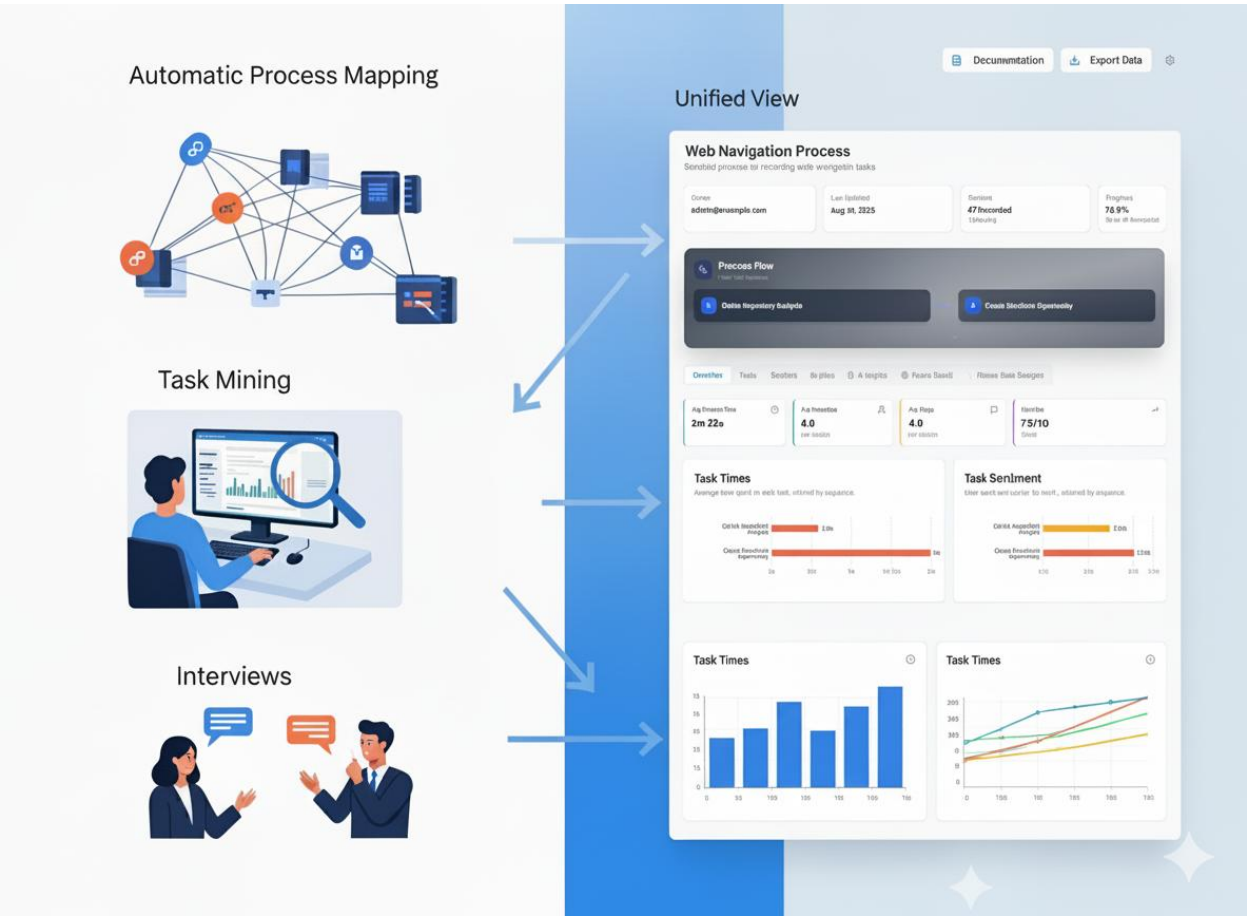
- Threat model early
    - Identify assets (PII, financial data), actors, attack surfaces (integrations, agents).
    - Prioritize with STRIDE or similar; define mitigations and test plans.
- Regulatory constraints
    - Map applicable regs (e.g., HIPAA, CJIS, GDPR/CCPA) to control requirements.
    - Set retention, auditability, and access logging policies; test them.
- AI risk boundaries
    - Define allowable use cases and prohibited data for LLMs/agents.
    - Configure tool/permission boundaries, human-in-the-loop checkpoints, rate limits, and a kill-switch.
    - Maintain prompt/version control, evaluation sets, and drift monitoring.
- Change & incident processes
    - Integrate CAB/CCB with risk scoring for changes.
    - Incident runbook: severity matrix, comms tree, containment/eradication/recovery steps, post-mortem template.
- Evidence collection
    - Automate control evidence where possible (access logs, approvals, test results).
    - Keep a controls register mapped to audits; schedule periodic reviews.

Action Steps: Scope canvas; governance charter; stakeholder heatmap; benefits baselining; risk register kickoff.

Artifacts: OKR (objectives and key results) alignment grid; Governance RACI; Change Control policy; Data readiness checklist; Benefits Realization plan.

---

3) Discover the Current State (Fact-based)



Objectives: Create a **single source of truth** for how work really gets done using multiple lenses.

Discovery Lenses:

- **Automatic process mapping & task mining** (ClearWork browser & desktop capture: URLs, clicks, forms, timestamps) for user-level workflows.
    - What it is: Using ClearWork's browser capture to log real user activity (URLs, clicks, field inputs, timestamps) in sequence.
    - Best practices:
        - Scope first: Choose high-value flows (e.g., order entry, case resolution, invoice approval). Don't boil the ocean—capture 2–3 critical journeys per function to start.
        - Consent & privacy: Inform users about capture, mask sensitive fields, and follow least-privilege rules.

        - Capture duration: Run 2–4 weeks of capture to smooth out variance (seasonality, shifts).
        - Annotate events: Tag meaningful events (submit order, close case, escalate) so they can be aligned with business outcomes.
    - Output: Heatmaps of time-on-task, sequences of clicks/forms, exception rates.
    - Pitfalls to avoid: Over-capturing (too many flows at once), ignoring edge cases, failing to anonymize sensitive data.
- **Structured mapping** using **BPMN 2.0** and **Value Stream Mapping** for clarity and waste spotting.
    - What it is: Converting captured steps into standardized diagrams and value stream maps.
    - Best practices:
        - BPMN 2.0
        - Use consistent symbols (start/end, gateways, swimlanes for roles).
        - Highlight exception paths separately to avoid clutter.
        - Maintain a repository so maps stay versioned and accessible.
    - Value Stream Mapping (VSM)
        - Plot process stages with cycle time, wait time, % complete/accurate.
        - Distinguish value-added vs. non-value-added time to spot waste.
        - Add metrics (WIP inventory, handoffs, SLA).
    - Output: Visual “current state” map showing bottlenecks, waste, and ownership.
    - Pitfalls: Maps that are too detailed (over-engineered spaghetti) or too shallow (just boxes without times/metrics).
- **Qualitative** (interviews, shadowing, VOC/VOE, surveys); triangulate with telemetry.
    - What it is: Capturing context and perception to complement telemetry.
    - Best practices:
        - Interviews/shadowing:
            - Use structured guides (what tasks take longest? where are the workarounds?).
            - Observe actual behavior vs. reported behavior.

- Voice of the Customer (VOC):
    - Gather customer feedback linked to process stages (e.g., onboarding, support).
- Voice of the Employee (VOE):
    - Survey pain points (duplicate entry, unclear approvals).
    - Encourage open "friction stories" to capture nuance.
- Triangulation: Compare qualitative findings with telemetry (e.g., users say system is slow; logs show 30% of time spent waiting for page loads).

- Output: Insights into culture, motivation, and hidden obstacles.
- Pitfalls: Treating anecdotes as universal truth; ignoring triangulation.

- **Controls & Compliance** (segregation of duties, SLA breaches, policy exceptions).
    - What it is: Identifying risks, breaches, and control weaknesses embedded in current workflows.
    - Best practices:
        - Segregation of duties (SoD): Map roles to tasks to detect toxic combinations (e.g., same user creates and approves invoice).
        - SLA breaches: Capture step durations vs. contractual/operational SLAs.
        - Policy exceptions: Log manual overrides, skipped steps, approvals done outside the system.
        - Audit readiness: Tag steps with compliance relevance (financial, security, privacy).
    - Output: Control map highlighting risk hotspots.
    - Pitfalls: Over-focusing on formal controls and ignoring informal workarounds.

Baseline & Insights:

- Throughput, lead times, error/rework, handoff delays, queue times, cost to serve.
    - Best practices:
        - Throughput: Volume of items completed (e.g., invoices/day).
        - Lead time: End-to-end time from request to completion.
        - Cycle time: Time spent actively working vs. waiting.

- Rework/error rate: % of cases re-opened or sent back.
- Handoff delays: Average delay between role transitions.
- Queue times: WIP aging—items waiting > SLA.
- Cost to serve: Labor + system cost per transaction.

  - Output: A baseline performance dashboard.
  - Pitfalls: Using averages only; hide variability. Always track distributions.

- Friction log: repetitive steps, swivel-chairing, duplicate data entry, unclear ownership.
  - Examples:
    - Repetitive steps – entering the same data into multiple systems.
    - Swivel chairing – copying/pasting between apps.
    - Duplicate entry – redundant approvals, forms.
    - Unclear ownership – tasks left in limbo.
  - Best practices:
    - Crowdsource via surveys + confirm with telemetry.
    - Quantify time/effort lost per friction item.
    - Prioritize by impact × frequency.
  - Output: Prioritized friction log linked to business outcomes.
  - Pitfalls: Treating all frictions as equal; not validating anecdotes with data.
- System map: application landscape, integrations, identity boundaries, data sensitivity.
  - Best practices:
    - Applications: List every system used in the flow, purpose, owner.
    - Integrations: Document interfaces (APIs, ETL, file drops).
    - Identity boundaries: Note where users re-authenticate; highlight weak SSO coverage.
    - Data sensitivity: Tag systems with PII, PHI, financial data.
    - Dependencies: Upstream/downstream systems (what breaks if this goes down).
  - Output: Clear map for IT and governance bodies.

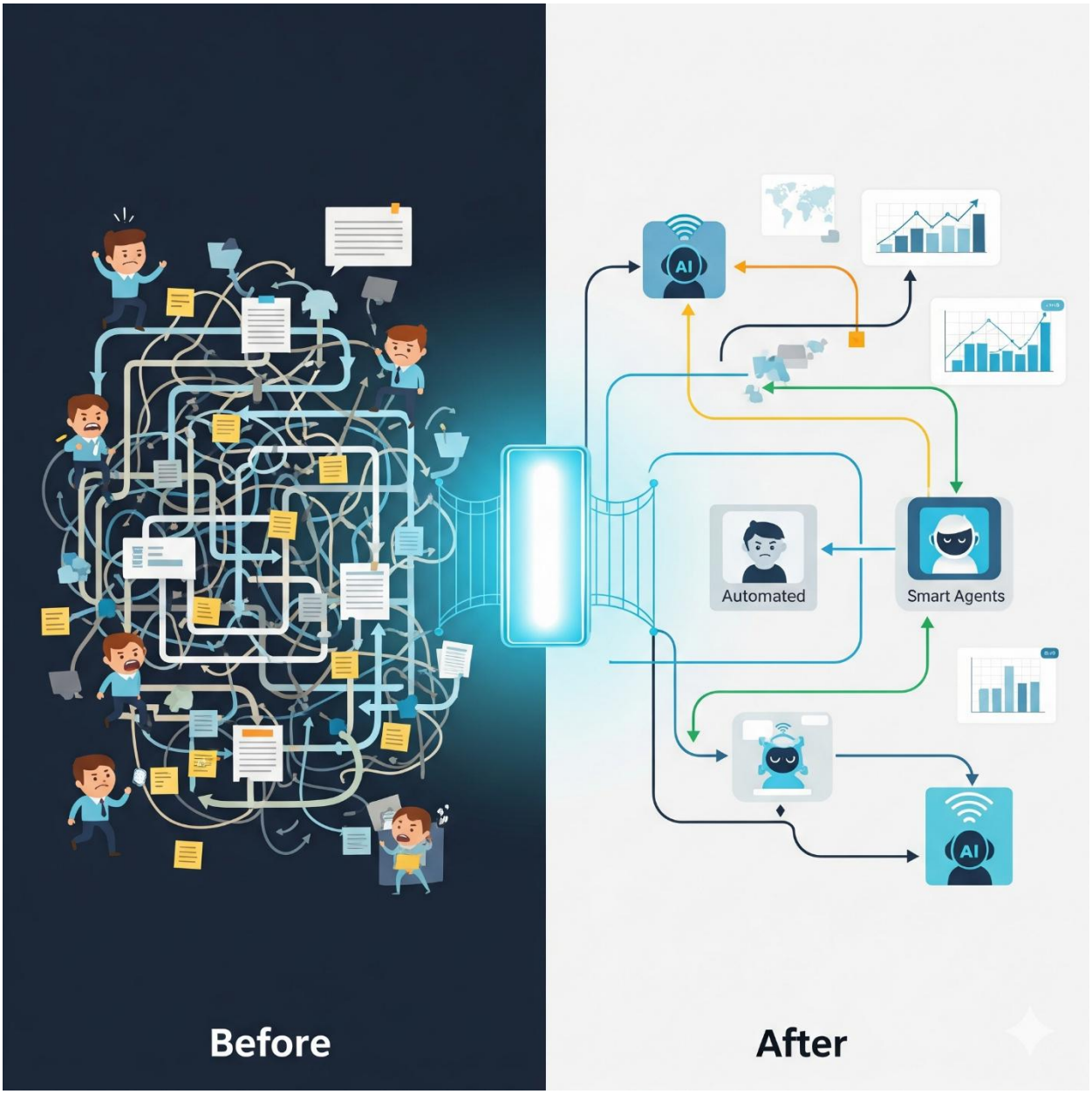
- Pitfalls: Missing “shadow IT” tools (Excel macros, shared drives, SaaS point tools).

Action Steps: Deploy ClearWork capture; run discovery sprints; hold playback sessions; confirm with SMEs.

Artifacts: Current-state BPMN; value stream map; heatmap of bottlenecks; friction log; system & data lineage map.

---

4) Design the Future State (Process First, Tech Second)



Objectives: Redesign for flow, quality, and measurability. Decide what to **eliminate, simplify, standardize, automate, or augment with AI/agents**.

Design Principles:

- Start with **outcomes** and guardrails; codify **standard work** and exception paths.
    - Anchor in outcomes:

        - Link every process redesign to business KPIs (e.g., reduce claims cycle time from 14 → 7 days).
        - Define explicit value hypotheses for each change.
    - Guardrails:
        - Define boundaries up front (privacy, uptime SLOs, SLA requirements).
        - Use “cannot violate” rules (e.g., PII cannot leave EU data centers).
    - Standard vs. exception:
        - Codify the 90% path as the standard workflow.
        - Document exception paths (escalations, approvals, error handling) separately for clarity.
    - Best practice: Standard work is the default path in tools — exceptions require explicit triggers.
- Design for **human-in-the-loop** and **least privilege**; make controls observable.
    - Human in the loop (HITL):
        - Define decision checkpoints where human judgment adds value (e.g., fraud detection thresholds).
        - Use tiered approvals (auto-approve < $1K; manual > $1K).
    - Least privilege:
        - Apply RBAC and service accounts with scoped access.
        - Automations/agents only get the minimal permissions needed.
    - Observable controls:
        - Make approvals, overrides, and escalations visible in dashboards and logs.
        - Provide audit trails for all exception handling.
- Prefer **platform patterns** (workflow, rules, integration, data products) over point bots.
    - Why platform patterns:
        - Workflow engines, integration platforms (iPaaS), decision rules engines, and data product platforms scale and are maintainable.
        - Point solutions (isolated RPA bots, macro scripts) create brittle dependencies but at the same time can deliver quick wins. Try to

standardize on technologies (such as RPA providers, integration tools, etc.).

    - Best practices:
        - Use workflow platforms for orchestrating processes end-to-end.
        - Use rules engines for decisions that frequently change.
        - Standardize on integration hubs to reduce spaghetti APIs.
    - Pitfall: Starting with tactical bots for "quick wins" and then failing to scale.
- Build for **changeability** (versioned SOPs, modular automations, test harnesses).
    - Versioned SOPs:
        - Treat SOPs like code — version them, track history, and update with every change.
    - Modular automations:
        - Break workflows into reusable modules (login, submit form, validate data).
        - Enables swapping components without breaking the whole.
    - Test harnesses:
        - Automated regression tests for workflows and agents.
        - Synthetic test data for sensitive use cases.
    - Best practice: Build a "design-for-change" mindset — assume the process will evolve in 6–12 months.

Automation & AI/Agents:

- Build an **Automation Candidate Heatmap** (frequency × effort × error rate × cycle time × risk × data access).
    - Scoring dimensions:
        - Frequency (times per week/month).
        - Effort (hours per instance).
        - Error rate (% of cases reworked).
        - Cycle time impact (how long it holds up the process).
        - Risk (financial, regulatory, reputational).
        - Data access (are APIs available; is data structured).

- Output: A prioritized matrix showing high-value candidates for automation.
- Best practice: Use both quantitative data (ClearWork logs) and qualitative input (employee pain points).

- Choose the right tool for the job: workflow/BPM, iPaaS, RPA, decisioning, search, or **agentic orchestration**.
  - Workflow/BPM: End-to-end orchestrations across teams.
  - iPaaS (integration platforms): System-to-system data sync.
  - RPA: UI-level automation when APIs are unavailable
  - Decisioning engines: High-volume rules (e.g., loan approvals).
  - Search/knowledge tools: Empower users with retrieval instead of repeating queries.
  - Agentic orchestration: LLM-driven agents coordinating multi-step tasks across tools.
  - Best practice: Default to platforms → fall back to bots only where APIs don't exist.
- Define **agent boundaries** (tools, permissions, rate limits, kill-switch, logs) and observability.
  - Boundaries:
    - Explicitly list tools accessible, permissions, and scope of tasks.
    - Set rate limits to avoid runaway loops.
    - Create a kill-switch to shut down an agent instantly if needed.
  - Observability:
    - Log every step, prompt, and action.
    - Provide a replay mode for debugging.
    - Alert when agents drift outside normal patterns.
  - Best practice: Treat agents like new employees → onboard them with SOPs, supervise, then grant autonomy gradually.
- LLMOps basics: prompts as code, versioning, evaluation sets, drift monitoring, feedback loops.
  - Prompts as code: Store prompts in Git; version-control them like software.
  - Versioning: Tag model + prompt + context → reproducible outputs.

- Evaluation sets: Maintain test cases for expected agent responses.
- Drift monitoring: Watch for accuracy degradation as data or models change.
- Feedback loops: Allow users to rate responses, and use that data to retrain/refine.
- Best practice: Apply the same rigor as ML Ops to LLM agents.

Governance Gates: Architecture review; security & privacy assessment; change control; value check.

- Key gates:
  - Architecture Review
    - Ensure solution aligns with enterprise standards and guardrails.
    - Validate scalability, integration patterns, and reuse.
  - Security & Privacy Assessment
    - Validate access controls, data handling, and compliance with privacy regs.
    - Check that AI agents respect data boundaries and logging.
  - Change Control (CAB/CCB)
    - Review risk and backout plans for production changes.
    - Require impact analysis on end-users and downstream systems.
  - Value Check
    - Confirm automation delivers measurable value vs. hypothesis.
    - Kill/adjust initiatives that do not meet value targets.
- Best practice: Keep gates lean but firm — use checklists and templates to make reviews fast but consistent.

Action Steps: Co-design target BPMN; draft SOPs; run value/APT (assumption-based testing); simulate throughput; plan cutover options.

Artifacts: Future-state BPMN/SOPs; Automation Heatmap; AI/Agent safety spec; test plan; change impact assessment.

---

5) Plan & Deliver (Incremental Value)

Objectives: Deliver value in short, auditable slices with explicit adoption plans.

Delivery Plan:

- Wave planning (90-day horizons), with monthly value checkpoints; include de-risking spikes.
    - • Why 90 days? Long enough to deliver meaningful value; short enough to maintain urgency and adaptability.

- Structure:
    - Wave 1: High-confidence, low-dependency wins (to build momentum).
    - Wave 2: Medium-dependency workstreams (integration-heavy, cross-functional).
    - Wave 3+: Complex initiatives with cultural change or high compliance risk.
  - Monthly value checkpoints:
    - At the end of each month, validate whether benefits are emerging.
    - Adjust scope mid-wave if metrics are off-track.
  - De-risking spikes:
    - Allocate time from each wave to run "spikes": short experiments that resolve unknowns (API limits, data privacy checks, agent safety tests).
- Backlog triage using **RICE** scoring (Reach × Impact × Confidence ÷ Effort).
  - Formula: Reach × Impact × Confidence ÷ Effort.
    - Reach: How many users/processes are affected?
    - Impact: How much does it move the KPI needle? (High/Medium/Low, mapped to %).
    - Confidence: How certain are we in our assumptions (data-backed vs. anecdotal)?
    - Effort: Estimated complexity (time, resources, cost).
  - Execution:
    - Score each backlog item with team input.
    - Rank-order automatically by RICE score.
    - Revisit monthly — scores change as new data emerges.
  - Best practice: Keep scoring consistent by pre-defining ranges (e.g., "Reach = # of users impacted per month").
- **Readiness gates:** design complete, data ready, security controls in place, training content ready, support model confirmed.
  - Purpose: Ensure nothing goes live until critical foundations are in place.
  - Checklist:

- Design complete → Future-state SOP/BPMN validated with SMEs.
- Data ready → Clean, integrated, permissions approved.
- Security controls in place → RBAC, audit logs, masking tested.
- Training content ready → Role-based job aids, FAQs, quick reference guides.
- Support model confirmed → Hypercare owners, escalation paths, SLA response times.

  - Best practice: Gates are binary → pass/fail. No partial credit.

- **Testing:** Unit, integration, UAT, security/privacy tests, red-team for agents; rollback plans.
  - Types of tests:
    - Unit testing: Validate each workflow/automation step in isolation.
    - Integration testing: Confirm handoffs across systems and teams.
    - User Acceptance Testing (UAT): End-users validate usability, outcomes, and exception handling.
    - Security & privacy tests: Validate encryption, masking, access logging, retention rules.
    - Red-team testing for agents: Simulate malicious prompts, overloads, or drift to test resilience.
  - Rollback plans:
    - Document a step-by-step backout plan for every change.
    - Pre-validate that reverting to prior state is technically possible.
  - Best practice: Treat red-team testing as standard for AI/agent use if possible.
- **Cutover:** Dark launch, canary, phased region/department rollouts.
  - Strategies:
    - Dark launch: Deploy in production but hidden from users → test telemetry before exposure.
    - Canary release: Roll out to a small % of users or a single department. Monitor adoption and errors.
    - Phased rollout: Expand by geography, role, or function (e.g., start with APAC before global).
  - Key steps:

- Monitor telemetry and user sentiment continuously.
- Set rollback trigger thresholds (error % > 5, SLA breach > 10%, adoption < 30%).
- Keep hypercare team on call during early rollout.

  - Best practice: Never flip a full switch — always layer risk with staged exposure.

- **Outputs / Deliverables**
  - Wave roadmap (90-day slices with milestones and checkpoints).
  - Prioritized backlog with RICE scores.
  - Readiness gate checklist signed off by owners.
  - Test evidence log (unit, integration, UAT, security, red-team).
  - Cutover plan with dark launch/canary strategy, rollback triggers, and hypercare schedule.

Action Steps: Build an integrated plan (TO); create runbooks; pre-brief CAB/CCB; schedule SteerCo value reviews; instrument telemetry.

Artifacts: Integrated release plan; RICE matrix; test evidence; go-live checklist; hypercare playbook.

---

6) Drive Adoption & Realize Benefits

Objectives: Make the new way of working **the easy way**—and prove the value.

People & Adoption:

- **ADKAR** plan (Awareness, Desire, Knowledge, Ability, Reinforcement), mapped to personas.
    - Awareness: Communicate the why behind the change. Use tailored messages per persona (execs hear about ROI, frontline hear about time saved).

- Desire: Involve employees early (pilot programs, beta access). Highlight personal benefits (“less rework,” “fewer screens to click through”).
- Knowledge: Provide clear, concise learning content: role-based guides, quick videos, job aids. Make training searchable and accessible on demand.
- Ability: Ensure users can apply the change: practice sessions, sandbox access, live coaching.
- Reinforcement: Recognize and reward adoption. Leaders model behavior; managers use dashboards to track usage.
- Best practice: Map ADKAR to each persona (e.g., Sales Rep, Finance Analyst, Support Agent) so communications and enablement fit context.

- Sponsor roadmap; champions network; role-based enablement; job aids embedded in-flow.
  - Sponsor roadmap: Pre-schedule sponsor moments — launch kickoff, wave reviews, success celebrations. Sponsors must be visible (videos, town halls).
  - Champions network: Identify early adopters in each department. Give them early access, training, and recognition. Use them for peer-to-peer influence.
  - Role-based enablement: Build enablement content around specific roles (e.g., “How an AP clerk closes invoices in the new process”). Avoid generic “system training.”
  - Job aids embedded in flow: Put guidance where the work happens — in-app prompts, ClearWork nudges, hover tips, or side-panel SOP steps.
  - Best practice: Champions are not just trainers — they are advocates who shape adoption culture.
- Feedback loops: office hours, in-app prompts, “report a snag,” NPS/CSAT for internal users.
  - Office hours: Weekly drop-in sessions with SMEs and champions. Encourage safe sharing of “snags” or frustrations.
  - In-app prompts: Contextual nudges (“Need help? Click here to see the new SOP step”).
  - “Report a snag” feature: Lightweight feedback mechanism directly inside workflows. Route issues to a triage queue.
  - NPS/CSAT for internal users: Short (1–2 question) surveys tied to major process changes. Use results to guide refinements.
  - Best practice: Close the loop — show users their feedback resulted in improvements (“You said X, we fixed Y”).

Measurement:

- Before/after on the benefits register; leading indicators per wave.
    - Benefits register: Compare actuals vs. baseline for KPIs defined in readiness (e.g., cycle time, SLA attainment).
    - Leading indicators: Monitor early signals of adoption/value (e.g., % of transactions using new workflow, time-to-complete training, number of exceptions).
- Engineering: **DORA** metrics (deployment frequency, lead time, MTTR, change failure rate).
    - Deployment frequency: How often new automations/agents are released.
    - Lead time: Time from idea → production release.
    - MTTR (Mean Time to Recovery): How fast failures are resolved.
    - Change failure rate: % of changes causing incidents.
    - Best practice: Use DORA metrics to measure team performance and reliability of delivery.
- Ops: throughput, cycle time, right-first-time, queue depth, SLA attainment, cost-to-serve.
    - Throughput: Volume processed (e.g., invoices/day).
    - Cycle time: Duration from start to finish.
    - Right-first-time: % of cases completed without rework.
    - Queue depth: Work-in-progress aging — items waiting in backlog.
    - SLA attainment: % of items meeting service-level targets.
    - Cost to serve: Cost per unit transaction.
    - Best practice: Link ops metrics directly to OKRs so they resonate at the business level.

Reinforcement:

- Update SOPs; update incentives/OKRs; retire legacy steps; celebrate wins; publish “value stories.”
    - Update SOPs
        - Keep SOPs living documents. Every process improvement must result in an updated SOP within 48 hours of sign-off.
    - Update Incentives/OKRs

        - Align incentives with new behaviors. Example: support agents measured on % cases resolved using new workflow, not just volume.
    - Retire Legacy Steps
        - Remove access to obsolete tools/forms. Avoid “dual-running” indefinitely, which drains adoption.
    - Celebrate Wins
        - Publicly recognize teams who hit adoption and benefits milestones. Share user quotes.
    - Publish Value Stories
        - Show “before/after” case studies internally (time saved, errors reduced, happier customers).
        - Use data + narrative to make adoption emotionally rewarding.
    - Best practice: Treat reinforcement as continuous — not a one-off celebration.

Artifacts: Adoption dashboard; benefits tracker; persona-based enablement kit; SOP update log.

---

7) Continuous Improvement & Scale

Objectives: Institutionalize learning and extend transformation to adjacent processes.

Mechanisms:

- **PDCA** (Plan-Do-Check-Act) cadences at team and SteerCo; quarterly value reviews.
    - Team Level (weekly/biweekly):
        - Plan: Identify small improvements, assign owners.
        - Do: Implement on a limited scale (single squad, one system).

        - Check: Review metrics (throughput, error rate, adoption).
        - Act: Standardize if successful; discard if not.
    - SteerCo Level (quarterly):
        - Conduct value reviews against the benefits register.
        - Decide whether to scale, pivot, or stop initiatives based on data.
        - Refresh OKRs to reflect new opportunities.
    - Best practice: Keep PDCA cycles lightweight — 1-page update templates; dashboards auto-fed by telemetry.
    - Pitfall: PDCA treated as bureaucracy → focus shifts to "reporting" instead of experimentation.
    - Output: PDCA logs at team level, quarterly value review pack for SteerCo.
- Idea intake & triage; experiment backlog; A/B testing where feasible.
    - Idea intake:
        - Provide a simple channel (ClearWork feedback, form, Teams/Slack bot) for employees to submit pain points and ideas.
        - Encourage frontline users to suggest improvements — they see inefficiencies first.
    - Triage:
        - Categorize by effort/impact, compliance risk, and alignment to OKRs.
        - Reject or defer ideas quickly to avoid backlog bloat.
    - Experiment backlog:
        - Maintain a separate backlog for experiments distinct from core delivery.
        - Timebox experiments (1–4 weeks).
    - A/B testing:
        - Use structured pilots (Group A on current process, Group B on new design).
        - Compare cycle time, error rate, satisfaction.
    - Best practice: Treat failed experiments as wins if they prevent wasted investment.
    - Pitfall: Letting backlog grow without triage → creates idea debt.

- Output: Experiment backlog board (with owner, hypothesis, timebox, result).

- Technical debt and process debt backlogs; renewal plans.
  - Technical debt backlog:
    - Log shortcuts, workarounds, outdated integrations, manual patches.
    - Tag debt items with severity (low/medium/high risk).
  - Process debt backlog:
    - Document outdated SOPs, redundant approvals, "legacy" reports no one uses.
    - Measure cost of maintaining vs. eliminating.
  - Renewal plans:
    - Schedule quarterly "debt sprints" to retire accumulated technical/process debt.
    - Tie debt reduction to OKRs ("reduce manual touchpoints by 20%").
  - Best practice: Treat debt as visible work with owners and deadlines — not invisible "nice to have."
  - Pitfall: Letting debt accumulate until it blocks innovation.
  - Output: Unified backlog for tech + process debt, reviewed quarterly.
- COEs: Process Excellence, Automation/AI, Data/Analytics; communities of practice.
  - Purpose: Provide shared expertise and scaling mechanisms.
  - Process Excellence COE:
    - Maintains process maps, SOP standards, BPMN/VSM templates.
    - Owns training on process mapping and improvement.
  - Automation & AI COE:
    - Curates automation/agent standards, prompt libraries, safety guardrails.
    - Runs enablement for bot builders, workflow designers, and AI stewards.
  - Data & Analytics COE:
    - Manages data catalog, lineage, and KPI definitions.
    - Provides analysts to squads for discovery and value measurement.

- Communities of practice (CoPs):
  - Cross-functional user groups (e.g., “automation builders guild”) for peer-to-peer learning.
  - Encourage sharing of experiments, templates, failures.
- Best practice: Keep COEs enablers, not gatekeepers — provide guardrails and accelerators, not bottlenecks.
- Pitfall: COEs drifting into ivory tower mode, slowing delivery.
- Output: COE charters, community calendars, shared knowledge repositories.

Action Steps: Stand up CI rituals; publish a process health scorecard; schedule quarterly architecture reviews; maintain retirement roadmap for redundant apps.

Artifacts: CI backlog/Kanban; process health scorecard; experiment log; COE charters.

---

8) Governance, Risk & Compliance (Built-in, Not Bolt-on)

Objectives: Make good decisions fast—and safely.

Decision Bodies & Roles:

- **SteerCo:** Direction, funding, priorities, risk acceptance.
    - Purpose: Provide executive sponsorship and business alignment.
    - Accountabilities:

- Approve direction, funding, and scope changes.
- Prioritize competing initiatives.
- Accept or reject program-level risks.

- Cadence: Monthly or quarterly reviews.
- Membership: CIO/COO, CFO, Business Unit heads, Chief Data/AI Officer.
- Best practice: Keep meetings decision-focused → pre-read decks with 3 clear asks (funding, priority, risk).

- **Transformation Office (TO):** Cadence, transparency, unblockers, value tracking.
  - Accountabilities:
    - Set cadence of workstreams and reviews.
    - Ensure transparency (dashboards, benefits register).
    - Unblock delivery teams by escalating issues.
    - Track and report on value realization.
  - Cadence: Weekly working sessions, biweekly program syncs.
  - Membership: Program lead, process/automation leads, change/adoption lead, PMO rep.
  - Best practice: TO owns the "single source of truth" for progress and value tracking.
- **Architecture Review Board:** Standards, patterns, non-functional requirements.
  - Accountabilities:
    - Approve solution designs against standards.
    - Validate non-functional requirements (NFRs): scalability, security, resilience.
    - Promote reuse of platforms, patterns, and APIs.
  - Cadence: Biweekly reviews of major initiatives.
  - Membership: Enterprise architect, data architect, security architect, domain leads.
  - Best practice: Use lightweight checklists to speed reviews → no 50-slide decks.
- **Change Advisory/Control Boards (CAB/CCB):** Production change risk management.

- Accountabilities:
    - Classify changes (standard, normal, emergency).
    - Approve or reject production deployments.
    - Ensure rollback and comms plans exist.
  - Cadence: Weekly for normal changes, ad-hoc for emergencies.
  - Membership: Ops lead, service manager, security officer, business reps for high-impact systems.
  - Best practice: Automate evidence collection (test logs, sign-offs) to make CAB efficient.
- **Data/AI Governance Council:** Data stewardship, privacy, model risk, AI usage policies.
  - Accountabilities:
    - Define data stewardship responsibilities.
    - Approve policies on privacy, retention, lineage.
    - Review model risk assessments and agent governance boundaries.
    - Set policies for evaluation, monitoring, and incident handling.
  - Cadence: Monthly or as needed for AI/agent rollouts.
  - Membership: Chief Data Officer, Chief Privacy Officer, Legal/Compliance, AI/ML lead, business sponsor.
  - Best practice: Maintain a model/agent registry with approval status and risk classification.

Guardrails:

- Data classification & retention; access controls; audit trails.
  - Data classification: Label data (public, internal, confidential, restricted/PII/PHI).
  - Retention: Define retention and disposal rules (aligned to legal/regulatory).
  - Access controls: Apply RBAC and least-privilege across apps, automations, and agents.
  - Audit trails: Log all data access and changes; automate collection for audits.
  - Best practice: Classify data once, enforce rules everywhere (ClearWork, ERP, CRM, AI agents).

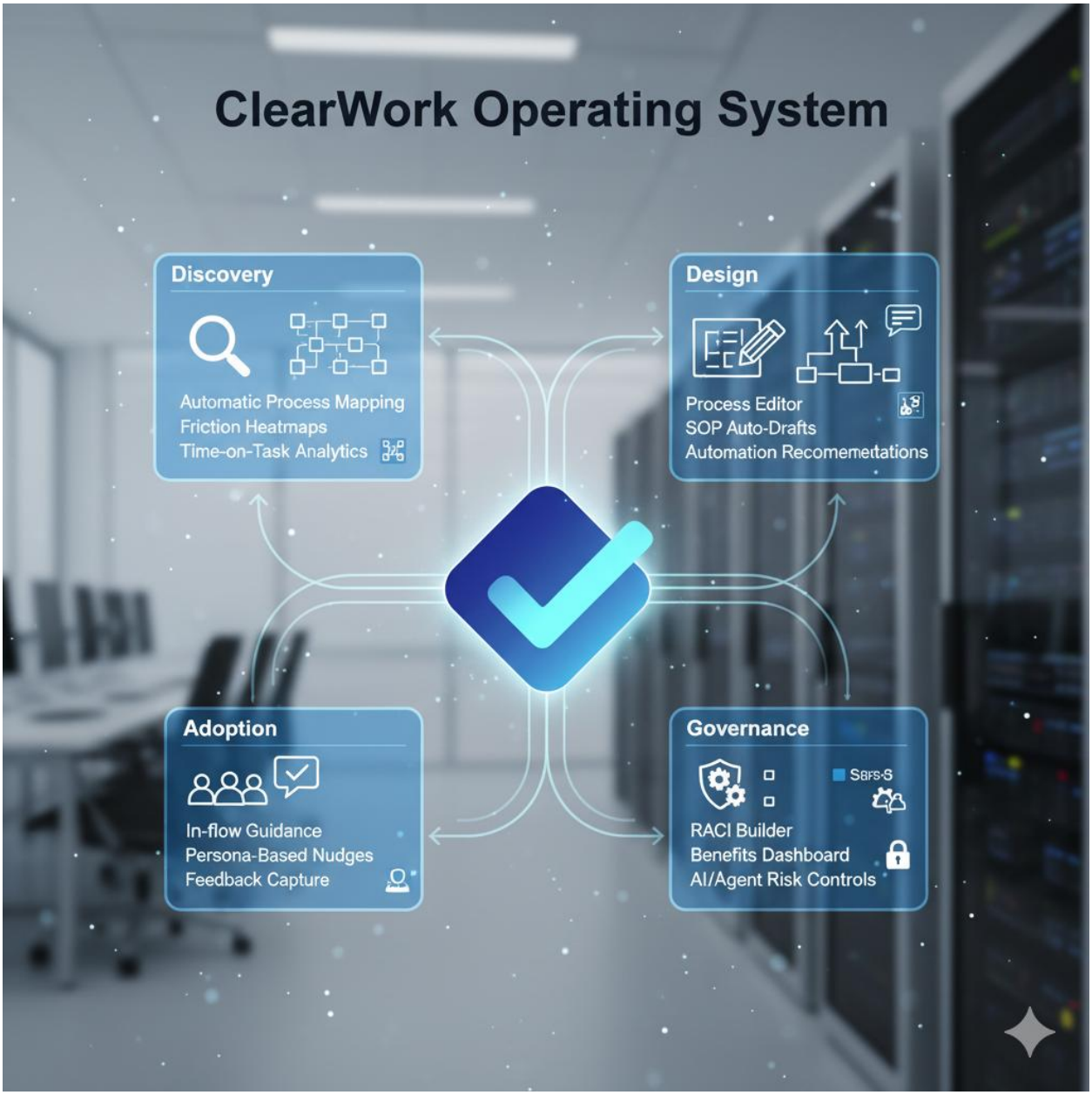
- **AI/Agent governance:** purpose limits, tool use boundaries, human-in-the-loop checkpoints, red-teaming, evaluation sets, drift monitoring, incident response.
    - Purpose limits: Define what agents can and cannot do (e.g., draft contracts yes, approve payments no).
    - Tool use boundaries: Limit agents to approved toolkits/APIs; no free exploration.
    - Human-in-the-loop checkpoints: Mandatory review for high-risk actions (approvals, financial postings, regulatory filings).
    - Red-teaming: Test agents against adversarial prompts and misuse scenarios.
    - Evaluation sets: Maintain test cases to validate consistency across model updates.
    - Drift monitoring: Watch for changes in accuracy or behavior as data/models evolve.
    - Incident response: Define escalation path, containment, and recovery for AI/agent failures.
    - Best practice: Treat AI agents like employees under probation → supervised, with increasing autonomy as they prove safe.
- Vendor & third-party risk; regulatory mappings.
    - Vendor risk: Assess SaaS/platform vendors for security posture, financial stability, SLA performance.
    - Third-party access: Limit and audit external user/system access.
    - Regulatory mappings: Maintain a controls matrix linking key regs (GDPR, HIPAA, SOX, PCI-DSS) to implemented policies and audits.
    - Best practice: Automate evidence (penetration test results, SOC 2 reports, access logs) and link it to governance reviews.

Action Steps: Approve governance charter; define decision SLAs; publish standards/guardrails; instrument compliance evidence collection.

Artifacts: Governance charter; decision matrix; standards catalog; AI/Agent policy; risk/controls register.

---

9) ClearWork Operating System (Where We Plug In)



This playbook lays out a step-by-step framework for transformation that avoids the pitfalls of traditional approaches. At its core, it is designed to be practical, measurable, and adaptive. It gives leaders the tools to define a value hypothesis, build the right governance spine, capture how work actually happens, design smarter workflows, embed automation and AI responsibly, and reinforce adoption through continuous improvement.

ClearWork accelerates this journey at every step. By automatically capturing user-level activity, generating process maps, surfacing friction points, and linking everything to adoption and governance dashboards, ClearWork provides the single source of truth that

transformation programs typically lack. It doesn't just map processes — it guides design, drives adoption, and embeds governance into daily workflows. With ClearWork, organizations move from abstract transformation promises to tangible, repeatable execution.

How ClearWork Helps at Every Phase

- **Discovery:** Automatic process mapping (user-level) with unified timeline; friction heatmaps; time-on-task analytics.
    - Challenge: Most transformations start with assumptions or interviews that only capture part of the truth. This leads to blind spots and flawed designs.
    - ClearWork Solution:
        - Automatic process mapping captures every click, form entry, and timestamp at the user level.
        - A unified timeline shows how work actually flows across systems and roles.
        - Friction heatmaps highlight bottlenecks like duplicate entry or swivel-chairing.
        - Time-on-task analytics quantify exactly where time and effort are wasted.
    - Impact: Leaders get a fact-based baseline to prioritize what matters most, replacing assumptions with hard evidence.
- **Design:** Process editor (BPMN-style), SOP auto-drafts, automation recommendations, agent boundary templates.
    - Challenge: Future-state processes are often drawn in workshops but rarely grounded in reality or optimized for automation/AI.
    - ClearWork Solution:
        - A BPMN-style process editor turns captured activity into structured workflows.
        - SOP auto-drafts reduce manual documentation effort.
        - Automation recommendations rank opportunities based on effort, frequency, and risk.
        - Agent boundary templates define safe scopes for AI use (tools, permissions, checkpoints).
    - Impact: Teams co-design processes that are practical, auditable, and automation-ready, while cutting down months of manual documentation.

- **Adoption:** In-flow guidance; persona-based nudges; ADKAR-aligned prompts; feedback capture.
    - Challenge: Even the best designs fail if employees don't adopt them. Traditional training and comms often feel disconnected from daily work.
    - ClearWork Solution:
        - In-flow guidance provides real-time prompts embedded in the applications users already work in and grounded on the actual usage of their system.
        - Persona-based nudges tailor adoption strategies for different roles.
        - ADKAR-aligned prompts guide users from awareness through reinforcement.
    - Impact: Adoption is built into the flow of work, making the new way the easiest way — and ensuring that value realization actually occurs.
- **AI Adoption:** How ClearWork Powers AI Adoption & Orchestration
    - Grounded in your data – ClearWork trains the agent on your organization's actual workflows, SOPs, and usage patterns, making outputs accurate, relevant, and trustworthy.
    - Context-aware guidance – The agent understands how your people really work (from captured activity), so its in-flow prompts, nudges, and recommendations are directly aligned to reality.
    - Cross-application reach – Unlike single-system copilots, ClearWork's agent spans across CRM, ERP, HRIS, ITSM, and custom apps — orchestrating end-to-end workflows
    - Agent-to-agent orchestration – Coordinates downstream specialized agents (e.g., finance bot, HR bot) by passing context, permissions, and data safely between them.
    - Safety built-in – Clear boundaries (permissions, logs, kill-switches, drift monitoring) ensure AI acts within defined guardrails, with humans in the loop where needed
    - Continuous learning – As processes evolve, the agent retrains on updated data and SOPs, staying in sync with how work actually happens.
    - Accelerates adoption – By being embedded in existing tools and automating repetitive steps, the agent makes the new way of working the easy way, driving natural adoption.
- **Continuous Improvement:** How ClearWork Supports Continuous Improvement

- Continuous tracking – continuously have users record activities to see how they are doing compared to expected results. Surface additional challenges to correct.
- Automated PDCA cycles – Links metrics to Plan–Do–Check–Act cadences at team and SteerCo levels, ensuring learning loops don't stall
- COE enablement – Provides templates, process libraries, and agent safety guidelines to empower Process, AI/Automation, and Data COEs.
- Communities of practice – Embeds best practices, value stories, and SOP updates into shared spaces so learning spreads across teams.
- Value tracking dashboards – Continuously compare actuals vs. baselines, making it easy to prove impact and prioritize the next improvement.

👉 **Bottom line:** With ClearWork, transformation stops being a one-off project that fades after go-live. It becomes a continuous, evidence-based operating model that improves how work is done, embeds automation and AI responsibly, and sustains adoption across the enterprise.
