## Problem Statement ✅ Refined Problem Statement Breakdown with Delivery Impact

**1. “Testing has no leadership to set clear vision, give comms, set standards of excellence.”**

**Details:** Testing is currently distributed across various BI, MI, and data projects without a centralized owner for defining strategy, tools, or processes.  
**Impact:** Without direction, teams work reactively, and testers rely on individual judgment. This leads to duplication of effort, slower alignment on priorities, and inconsistent stakeholder communication — increasing the delivery burden on project timelines.

**2. “Testing is individually led, so sporadic in its standards.”**

**Details:** Highly capable individuals drive testing within their domains, but practices vary greatly between teams and projects.  
**Impact:** Without shared standards, onboarding, handovers, and cross-project collaboration become inefficient. This inconsistency leads to higher testing effort, rework, and risk of quality gaps — all of which stretch limited resources and delay go-live timelines.

**3. “Exasperated when bringing in contractors.”**

**Details:** Contractors are often onboarded into siloed environments without reusable test artefacts, shared documentation, or standard frameworks.  
**Impact:** Ramp-up time is extended, and delivery velocity is impacted. Contractors may unintentionally reinvent processes or introduce variation, further reducing the efficiency of an already resource-constrained testing function.

**4. “Hard to get overall visibility of testing status.”**

**Details:** No unified dashboard or reporting process exists across testing efforts in data engineering or analytics delivery streams.  
**Impact:** Project leads and stakeholders lack clarity on readiness or blocker risks. Coordination becomes manual, and escalations arise late, adding pressure and affecting delivery predictability.

**5. “No permanent capability to support and progress test automation.”**

**Details:** Automation efforts are limited to individual testers, with no dedicated resource pool, roadmap, or framework in place.  
**Impact:** The testing team spends more time on repetitive manual validation. This reduces their ability to cover new test cases, increasing risk and slowing releases — particularly when timelines compress and resources are already stretched.

**6. “No defined Entry and Exit criteria.”**

**Details:** Testing often starts without structured scope definitions and ends without formal sign-offs.  
**Impact:** Without clear criteria, scope creep and misalignment on release readiness occur. This leads to extended test cycles, repeat testing, and rushed sign-offs — compounding resource pressure near go-live.

**7. “No governance of standards and processes.”**

**Details:** There is no single source of truth or enforced testing lifecycle governance.  
**Impact:** Best practices are inconsistently applied, resulting in fragmented QA documentation, uneven test coverage, and manual rework — which all demand high individual effort to compensate.

**8. “No QA checkpoints e.g. for release readiness.”**

**Details:** Projects lack embedded QA tollgates or standard readiness checklists before go-live.  
**Impact:** Testing teams are often under pressure to sign off without comprehensive verification, increasing the risk of post-release defects. This adds further burden on small testing teams to monitor production and deal with fallout.

**9. “Risk acceptance not defined.”**

**Details:** Projects move forward without documented QA risks or defect thresholds.  
**Impact:** When testing constraints are not escalated clearly, accountability is diffused and risk tolerance varies across projects. This can lead to uncontrolled go-lives and hidden defects, increasing workload downstream.

**10. “No warranty period monitoring.”**

**Details:** Post-release testing or defect trend analysis is not consistently tracked by the QA function.  
**Impact:** Issues are often detected late in production by end-users or business teams, putting unplanned pressure on testing teams to troubleshoot or rebuild validation efforts after go-live.

**🎯 Common Theme: Resource Constraints Driving High Effort & Bottlenecks**

Across all the above challenges, the root impact stems from:

* **High manual effort** due to lack of automation and standardisation
* **Isolated knowledge** not being shared across testers
* **No centre-led governance** to streamline repeatable quality practices
* **Delivery teams overrelying on few testers**, creating burnout and delays

Would you like this converted into a multi-slide **PowerPoint format** with icons/visuals and speaker notes to support your Data Engineering Lead’s funding case? I can include a “Proposed Solution – Testing CoE” slide next.

**You said:**

change - No defined Entry and Exit criteria - By project team to justfy the testing against it.

**ChatGPT said:**

Absolutely — here’s the **refined version** of that specific point, updated to highlight that the **project team lacks clear Entry and Exit criteria** to justify testing outcomes:

**🔹 Original:**

**“No defined Entry and Exit criteria.”**

**✅ Refined Version:**

**“No defined Entry and Exit criteria for the project team to justify the scope and completeness of testing.”**  
**Details:** Projects often proceed without mutually agreed test start/stop conditions or documented success thresholds.  
**Impact:** This leads to misaligned expectations between delivery and QA, causes test scope ambiguity, and results in unclear readiness signals. The lack of shared criteria puts added pressure on testers to validate "just enough" under tight timelines, increasing delivery risk in a resource-constrained environment.

Would you like me to reflect this change in the PowerPoint version as well?

**You said:**

change - No warranty period monitoring - Testing complete post go live , there is no dedicated tester pool to montior or support during the warrenty period monitoring

**ChatGPT said:**

Certainly — here is the **refined version** of that problem statement, updated to reflect the **absence of dedicated testing support during the post-go-live warranty period**:

**🔹 Original:**

**“No warranty period monitoring.”**

**✅ Refined Version:**

**“Testing completes at go-live, with no dedicated tester pool to support or monitor during the warranty period.”**  
**Details:** After deployment, there's no assigned QA ownership for early life support or issue triage during the warranty window.  
**Impact:** Post-release issues are identified reactively by business or production teams, placing unplanned pressure back on limited testers. Without structured monitoring, recurring issues may go untracked, and test coverage gaps may not be fed back into future sprints — resulting in delivery inefficiencies and potential reputational risk.

Would you like both of these updated points added into your presentation file or included in a formatted slide set?

**You said:**

For - Testing has no leadership to set clear vision, give comms, set standards of excellence.- Acrtually need head of testing role

**ChatGPT said:**

Great clarification — here's a **refined version** of that statement, explicitly highlighting the **need for a Head of Testing role** while preserving the original language and tone:

**🔹 Original:**

**“Testing has no leadership to set clear vision, give comms, set standards of excellence.”**

**✅ Refined Version:**

**“There is no Head of Testing in place to define a clear vision, lead communication, and establish standards of excellence across data testing.”**  
**Details:** In the absence of formal leadership, testing is fragmented across initiatives, with no central owner to guide strategy, tooling, quality metrics, or talent development.  
**Impact:** This gap limits the scalability and maturity of the testing function. Testers operate in silos without a shared direction, which results in inconsistent practices, limited alignment with delivery leadership, and missed opportunities for optimization. The lack of senior leadership makes it harder to advocate for investment in automation, governance, and CoE development — all of which are essential under current resource constraints.

### 10. ****Unclear Prioritisation Under Resource Constraints****

**Problem:** With limited testers, it’s often unclear which areas need deeper coverage and which can be risk-accepted.  
**Impact:** Critical paths might get missed or tested last, while low-priority items consume time. This leads to delayed go-lives or production defects.

### 7. ****Inconsistent Knowledge Sharing and Handover****

**Problem:** Lessons learned, best practices, and test artefacts are not captured centrally.  
**Impact:** Every tester reinvents the wheel. Contractors leave with critical context. Onboarding new resources takes longer and increases ramp-up cost.

### 2. ****No Standard Test Data Management Process****

**Problem:** Testers have to manually source, mask, or generate data, often with little support or tooling.  
**Impact:** High setup time for each project. Data inconsistency leads to flaky test results, increasing manual effort and slowing down cycles.

**✅ Vision, Mission, and Core Values – Testing Centre of Excellence**

**🌟 Vision**

To build a world-class Data Testing Centre of Excellence that enables high-trust, high-quality data products by embedding robust, repeatable, and scalable testing practices across all business intelligence, MI, and data engineering initiatives at Allianz UK.

**🎯 Mission**

To drive excellence in data testing through central leadership, standardisation, automation, and continuous improvement — ensuring every data-driven decision at Allianz UK is powered by verified, accurate, and reliable information.

**💠 Core Values**

(Aligned with Allianz Insurance UK values and principles)

1. **Customer & Data First**  
   We ensure the data behind every product, report, and decision is trustworthy and validated — protecting both the business and the customer.
2. **Quality through Accountability**  
   We own the integrity of every test cycle. Our work represents a commitment to consistent, reliable delivery, regardless of scale or complexity.
3. **Transparency & Governance**  
   We promote traceable, auditable, and measurable testing practices that reduce risk and support compliance.
4. **Collaboration & Respect**  
   We work closely with data engineers, analysts, and product owners, respecting domain expertise and encouraging shared ownership of quality.
5. **Continuous Learning & Innovation**  
   We invest in modern tools, automation, and skill-building to stay ahead of data and technology shifts.
6. **Efficiency through Standardisation**  
   We avoid duplicated effort by building reusable frameworks and centralising best practices, enabling delivery teams to scale testing faster and smarter.

#### **Mission**

To ensure the accuracy, quality, and reliability of data products and systems through rigorous, efficient, and innovative testing practices, enabling Allianz Insurance UK to make trusted data-driven decisions and deliver outstanding customer experiences.

#### **Vision**

To be a leading centre of excellence in data testing within Allianz UK – recognized for proactive risk mitigation, continuous quality assurance, and enabling high-performance data ecosystems that support sustainable growth, digital transformation, and operational excellence.

**1. Test Requirements**

**Purpose:** Define what needs to be tested and ensure traceability to business goals.

* 1.1 **Scope** – Identify the boundaries of what will be tested.
* 1.2 **Traceability** – Map requirements to test cases for coverage assurance.

**2. Planning**

**Purpose:** Prepare the testing team, tools, and timelines before execution begins.

* 2.1 **Resource allocation** – Assign testers and roles for project needs.
* 2.2 **Effort estimation** – Estimate time and capacity for each testing phase.
* 2.3 **Environment setup** – Ensure test environments are ready and aligned.
* 2.4 **Test data HL scope** – Define high-level data needs and sources.
* 2.5 **Output test plan** – Document the test strategy and approach.
  + Entry/Exit criteria – Clarify when testing can start and end.
* 2.6 **Number of test cycles** – Determine how many rounds of testing are required.
* 2.7 **Methodology decision** – Choose testing frameworks (e.g., Agile, Waterfall).

**3. Test Preparation**

**Purpose:** Build the foundations for consistent, repeatable test execution.

* 3.1 **Test design** – Write structured test cases or scenarios.
* 3.2 **Test data – creation** – Generate or mask data needed for test runs.
* 3.3 **Tooling prep / framework setup** – Configure tools and frameworks for execution.
* 3.4 **Peer review** – Validate test coverage and link scripts to requirements.

**4. Test Execution**

**Purpose:** Execute test cases and manage results, defects, and readiness.

* 4.1 **Smoke testing** – Perform quick checks to validate system readiness.
* 4.2 **Test cycle execution** – Run planned test cases across environments.
* 4.3 **Reporting and defect management** – Log, track, and report defects.
* 4.4 **Defect lifecycle** – Manage defect triage, retesting, and closure.
* 4.5 **Verify exit criteria** – Confirm all test goals have been met.
* 4.6 **Risk acceptance vs severity** – Evaluate unresolved issues and risks.
* 4.7 **Test readiness** – Validate that the system is ready for go-live.

**5. Release Readiness**

**Purpose:** Confirm release confidence and ensure post-go-live ownership.

* 5.1 **Test completion report** – Summarise execution results and defect metrics.
* 5.2 **Warranty contribution** – Prepare handover and support plan post-release.

**6. Regression & Automation**

**Purpose:** Build reusable test assets for efficiency and CI/CD alignment.

* 6.1 **Maintain automation packs** – Keep regression scripts updated for reuse.
* 6.2 **CI/CD integration** – Embed automated tests into the deployment pipeline.

**7. Process Improvement**

**Purpose:** Identify lessons learned and evolve testing quality.

* 7.1 **QA approach & self-audit** – Review the testing process and optimise with periodic checks.
* 7.2 **Documentation & knowledge base** – Keep testing documentation current and accessible.
* 7.3 **Test environment maintenance** – Manage and improve test environments, with a goal toward automation.

### ✅ ****Summary Workflow Diagram: End-to-End Automation Testing Lifecycle****

#### **1. Demand Intake & Opportunity Identification**

📥 Capture requests for automation from stakeholders or QA teams.  
🔎 Identify pain points: repetitive tasks, regression areas, long cycle times.

#### **2. Scope Assessment & Prioritization**

🧩 Evaluate effort vs. impact (ROI) for different test cases.  
🧮 Classify into: Feasible, Complex, Not Suitable.

#### **3. Feasibility Analysis & PoC (Proof of Concept)**

🧪 Run PoC on shortlisted use cases (data quality, schema, CDC, ETL).  
📊 Analyze success, tool compatibility, effort required.

#### **4. Tool Selection & Tech Stack Alignment**

🧰 Choose tool(s) based on data architecture, cloud infra, skill set.  
✅ Evaluate for CI/CD compatibility, connector support, alerting, scalability.

#### **5. Framework & Architecture Design**

📐 Define automation strategy: modular, reusable, plug-and-play.  
📁 Set naming conventions, folder structure, logging, and reporting setup.

#### **6. Automation Test Script Development**

💻 Build scripts for:

* Schema validation
* Null/dup checks
* Business rule validation
* Aggregate and metric checks
* Data reconciliation across layers

#### **7. Test Data Management**

📊 Create or source reusable test datasets (mock or production-anonymized).  
🔄 Enable parameterization for environment-agnostic runs.

#### **8. Integration with CI/CD & Scheduling**

🔁 Plug automation into pipelines (ADO, Jenkins, GitHub, etc.).  
⏰ Schedule via triggers (on data load, nightly, or event-driven).

#### **9. Execution & Monitoring**

👀 Monitor pass/fail, latency, job durations.  
📬 Enable alerting via email, Teams, Jira/Azure Boards.

#### **10. Reporting & Stakeholder Communication**

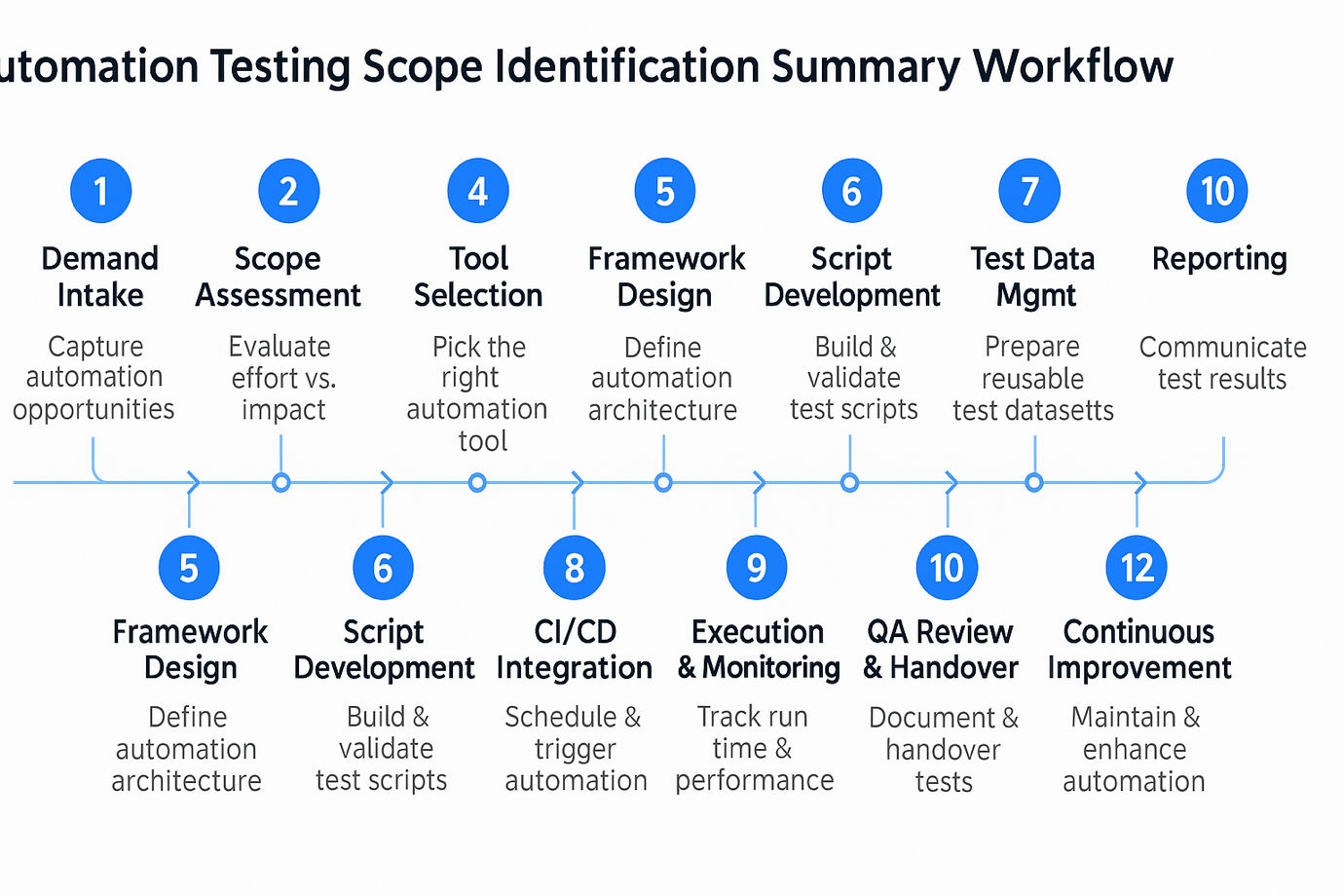
📈 Publish test execution summary, defects, trends.  
📄 Share dashboards and visualizations (e.g., Power BI, Grafana).

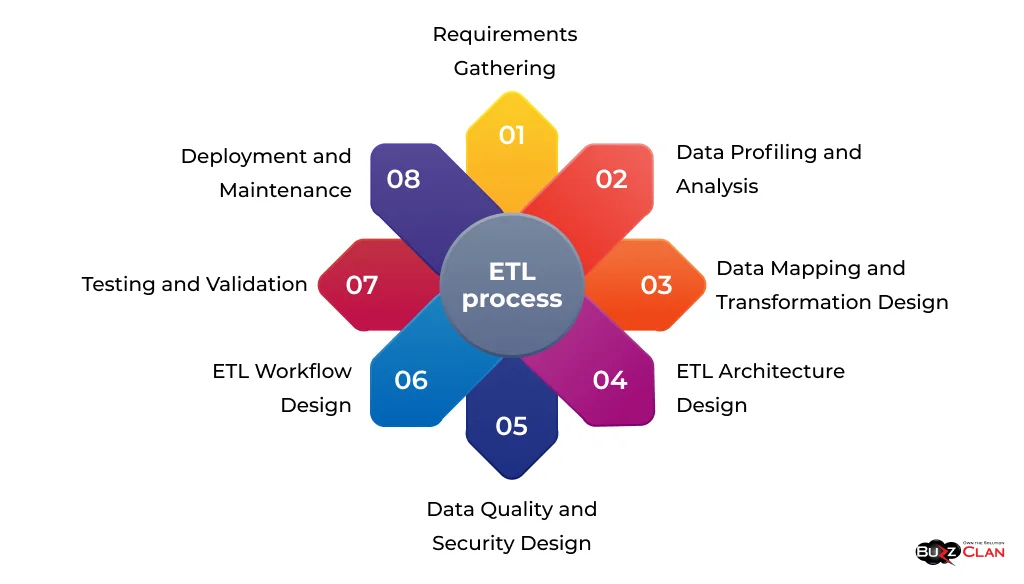
#### **11. QA Review, Documentation & Handover**

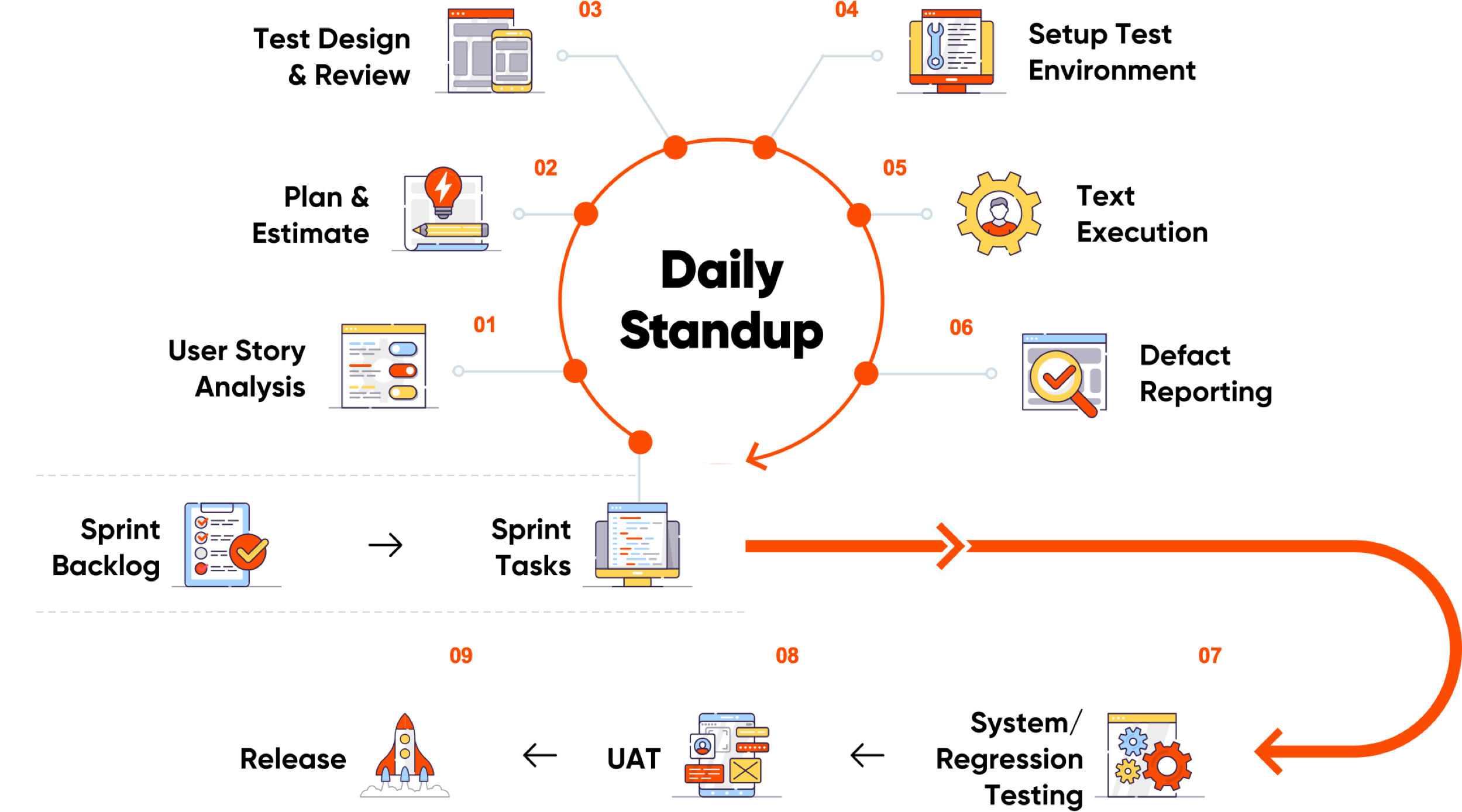
✅ Conduct peer review, code validation.  
📚 Maintain centralized documentation: test plans, coverage, SOPs.

#### **12. Maintenance & Continuous Improvement**

🔄 Update test cases with schema evolution, pipeline changes.  
📊 Track automation coverage, flaky test rate, regression impact.





--------

### ✅ ****Performance & Continuous Improvement – Data Testing****

This pillar ensures the Data Testing process evolves, stays efficient, and meets delivery SLAs through metrics, reviews, and enhancements.

#### 1. **Key Performance Metrics (KPIs)**

Monitor and optimize the following performance indicators regularly:

* ✅ **Test Case Execution Rate** (daily/weekly/monthly)
* 📉 **Defect Leakage Rate** (pre/post-production)
* ⏱️ **Average Execution Time per Test Suite**
* 📊 **Automation Coverage %** (manual vs. automated scope)
* 🛠️ **Time to Detect & Resolve Defects**
* 🔁 **Reusability Ratio** of test components
* 🧪 **Data Validation Accuracy %**
* 🧯 **MTTR** – Mean Time To Resolution for critical bugs
* 📅 **SLA Adherence** on testing turnaround times

#### 2. **Ongoing QA Reviews**

* 🔍 Periodic peer reviews of test cases and automation logic
* 🧪 Test effectiveness evaluation using retrospective defect trends
* 📋 Root Cause Analysis (RCA) workshops on high-severity bugs

#### 3. **Automation Audit & Optimization**

* 📦 Identify flaky or redundant scripts for optimization
* ♻️ Retire outdated checks and update to reflect schema/pipeline changes
* 📈 Introduce smarter validations (AI-assisted anomaly checks, sampling logic)

#### 4. **Environment & Tooling Efficiency**

* ⚙️ Monitor environment utilization and automation run times
* 🕵️ Detect and resolve environment drift, stale data, permission issues
* 🤖 Continuously upgrade frameworks and add new data connectors

#### 5. **Enhancement & Innovation Cycles**

* 💡 Quarterly review of feedback from QA, Dev, and Business to:
  + Add new automated checks (e.g., row-level rules, lineage)
  + Enhance alerting/reporting pipelines
  + Automate more regression/intelligence areas
* 📌 Document lessons learned and update SOPs accordingly

#### 6. **Knowledge Base & Capability Building**

* 📘 Maintain centralized repository of testing standards, guides, templates
* 🧠 Upskill QA team via focused learning sprints (e.g., PyTest, dbt test, Great Expectations)
* 🎯 Drive CoE-led hackathons or PoCs on evolving testing tools and AI integrations

### ✅ ****Testing Team Hierarchy & Responsibilities****

#### 1. **Head of Testing / QA Director**

* 🧭 **Responsibilities**:
  + Define testing vision, strategy, and budget
  + Align QA practices with enterprise data governance and business needs
  + Sponsor Testing Centre of Excellence (TCoE)
  + Manage QA risks and stakeholder reporting
  + Advocate for tooling, automation, and continuous improvement

#### 2. **QA/Test Manager (Manual & Automation Streams)**

* 🧰 **Responsibilities**:
  + Own planning, test strategy, and governance for project deliveries
  + Assign resources based on project complexity and timelines
  + Maintain test metrics dashboard and KPIs (e.g., defect leakage, execution rates)
  + Interface with Product Owners, Engineering Leads, and Business Analysts
  + Oversee defect triage, UAT support, and post-release validation

#### 3. **Test Automation Lead**

* 🤖 **Responsibilities**:
  + Define automation scope, standards, and framework governance
  + Evaluate and implement automation tools (e.g., PyTest, dbt, Great Expectations)
  + Maintain regression packs and CI/CD pipeline integration
  + Conduct code reviews and promote reuse of components
  + Drive PoCs and innovation in automation (e.g., anomaly detection, AI validation)

#### 4. **Senior Data QA Engineer**

* 🧪 **Responsibilities**:
  + Design and execute complex data validation scripts (SQL, Python)
  + Ensure end-to-end ETL/ELT coverage (Landing → Curation → Warehouse → Reports)
  + Create reusable test cases for CDC, schema change, reconciliation, lineage
  + Perform defect RCA and mentor junior testers
  + Liaise with DQ, metadata, and analytics teams for integrated testing

#### 5. **QA Engineers / Testers**

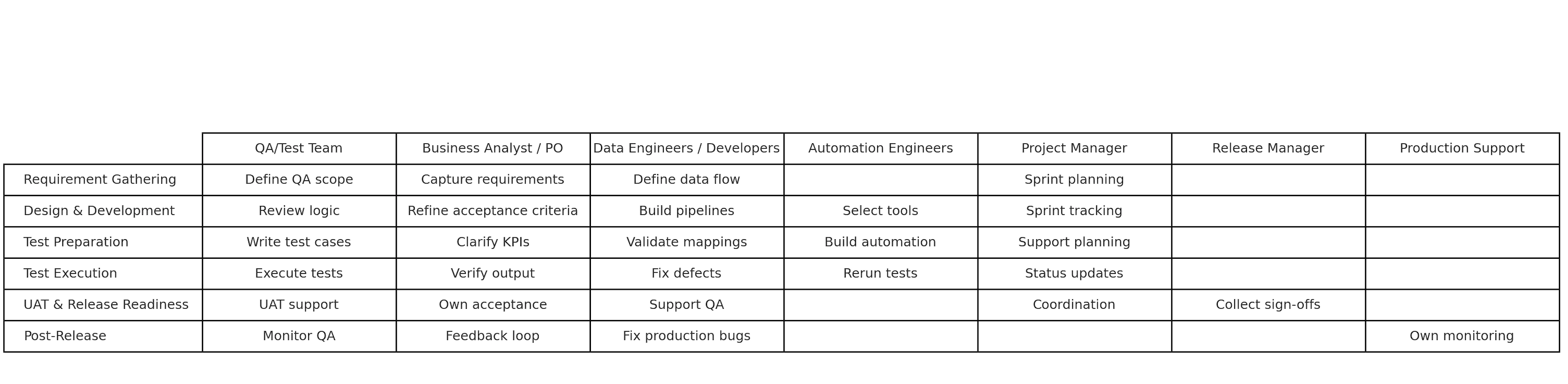
* 🛠️ **Responsibilities**:
  + Execute test cases based on business rules and technical mappings
  + Validate dashboards, filters, and KPIs against data marts
  + Perform smoke, regression, and exploratory testing
  + Raise and track defects in JIRA or similar tools
  + Maintain evidence and adhere to documentation protocols

#### 6. **UAT Coordinator / Business Tester (Optional)**

* 👥 **Responsibilities**:
  + Facilitate UAT planning and stakeholder involvement
  + Validate reports/insights against expected business outcomes
  + Collect sign-off and transition feedback to test/engg. teams

#### 7. **Environment & Release Coordinator (Shared or QA role)**

* 🌐 **Responsibilities**:
  + Ensure test environments are refreshed, accessible, and version-matched
  + Coordinate data setup, snapshots, and masking
  + Manage release cutover plans with delivery teams



RACI Matrix