Our Banking Client: shared QA platform

A payment-release story: 75 offshore QA staff, heavy manual testing and constrained test environments.

75

offshore QA staff

One payment release.
A platform the next team can reuse.

Executive briefing • Strategic vision • Client scenario

QE testing scenario: one timeout, two debits

PAY-142 simulates a CAD 100 transfer accepted before a timeout. The customer retries; an injected duplicate-posting defect tests whether QE catches the second debit.

Customer
Send CAD 100

→

Payment API
Key: pay-142

→

Provider
Accept; response times out

→

Customer retries
Reuse pay-142

Injected test defect

2 journals

Payer CAD 800
Recipient CAD 200

FAIL · duplicate posting

Expected test outcome

1 journal

Payer CAD 900
Recipient CAD 100

PASS · expected state

QE testing scenario. The injected defect causes the duplicate posting; a timeout alone does not imply two debits. Initial balances: CAD 1,000 and CAD 0. One journal contains one debit and one credit. No fees, FX or other activity.

Expected: one transfer, one balanced journal and one customer-visible result. This is a QE test scenario, not a reported client incident.

Outcomes beyond hours

01

Catch payment faults

Catch every critical seeded fault

Reviewed fault scenarios and independent balance assertions

02

Cover critical journeys

Exercise every critical scenario

AI drafts edge cases; domain QA approves the scenario matrix

03

Explain each release decision

Complete every decision record

CI captures evidence; AI drafts a source-linked summary

Supporting proof | Trust repeat runs | Validate real integrations | Enable the next squad

Baseline: Not recorded · Observed after: Not recorded · Targets proposed for pilot agreement.

Business aims: fewer escaped payment defects and emergency fixes. These proposed pilot measures do not establish those outcomes.

What AI contributes, from requirement to release

PAY-142 is an authored scenario. Each AI output needs review; test runners execute and people decide.

Before / after: separate modernization from AI

01 / Manual QE

Copy cases and run by hand
Shared data + scarce vendor slots
Reconcile screenshots and logs

02 / Modernized QE

Reviewed automated tests
Isolated data + virtual services
CI retains repeatable run evidence

03 / AI-assisted QE

Draft cases, code and fixtures
Suggest scope, diagnosis and repair
Summarize proof for human review

Establish the baseline

Active work · vendor waits · coverage gaps

Prove repeatability

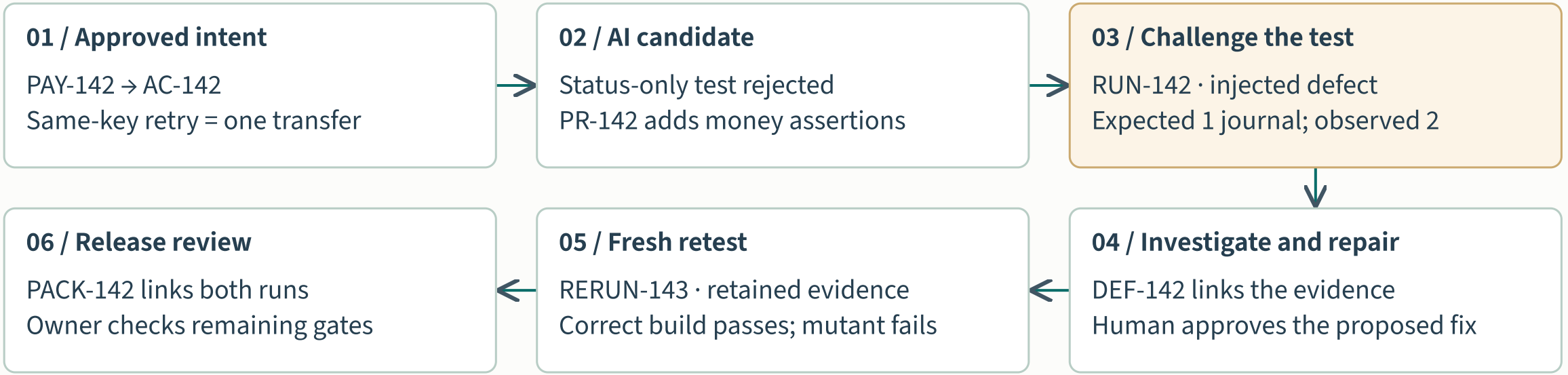
Environment readiness · reruns · raw evidence

Isolate added AI value

Candidate acceptance · review · rework

Same payment scope. The two target states are proposed; no productivity or quality improvement has been measured.

Follow one payment test through the handoffs



Illustrative artifacts and outcomes. Preserve failed evidence; a passing retest alone does not approve release.

Before / after: the shared QE architecture

BEFORE / SCENARIO BASELINE

Manual QE waits for test capacity

Payments QA | Web / mobile QA | Settlement QA

↓ Manual cases, reruns and evidence ↓

Few shared test environments

Backend services are not virtualized

Vendor test slots limit parallel runs

AFTER / PILOT TARGET

Squads reuse supported QE services

Payments QA | Web / mobile QA | Settlement QA

↓ Owned interfaces and reusable assets ↓

Shared QE platform

Context + fixtures + runners + evidence

AI drafts and explains · people review

QE modernization: isolated runs and controlled dependencies. **AI assistance:** preparation and interpretation.

Starting point: heavy manual QE and scarce vendor environments. Pilot target: controlled service tests, with real integration still required.

OUR BANKING CLIENT / ILLUSTRATIVE BANKING SCENARIO / EXECUTIVE

Four layers of AI-assisted QE

1

Assist a person

Individual copilots

Copilot drafts a retry test; QA reviews and runs it.

PROOF TO EXPAND

Useful, correct drafts after review

2

Assist a workflow

Connected workflow assistants

A CI assistant explains failures with links to evidence.

PROOF TO EXPAND

Traceable, useful recommendations

3

Delegate a bounded task

Bounded QE agents

An agent drafts a patch and reruns sandbox tests.

PROOF TO EXPAND

Repeatable actions within limits

4

Scale proven capability

Shared QE AI platform

Squads reuse context, evaluations, fixtures and evidence.

PROOF TO EXPAND

A second team can adopt and operate it

Shared foundations · start on day one Context · data · reliable tests · CI · QE modernization · owners

Layer 4 supports layers 1–3. Scale useful assistance without having to delegate more actions.

Authored roadmap. Expand with validated prerequisites and client evidence.

Adoption depends on funded foundations

SCOPE FOLLOWS PROVEN CAPABILITY

Inputs + people + evidence + approved AI access + a measured baseline

DRAFTING

Reviewed scenarios

Approved rules and observable outcomes.

↓

Domain QA accepts the scenario matrix.

Assess test design →

DIAGNOSIS ONLY

Evidence-linked triage

Retained failures, known dispositions and usable diagnostics.

↓

A reviewer checks the diagnosis; no retest runs.

Assess diagnosis →

EXECUTION

Reviewed test changes

Add repeatable environments, CI, fixtures, provider control and reliable runners.

↓

Independent assertions challenge the candidate.

Assess automation →

Our Banking Client has not been assessed. The dependency backlog can change scope, cost and timing; the 480-hour setup allowance is not a client quote.

OUR BANKING CLIENT / ILLUSTRATIVE BANKING SCENARIO / EXECUTIVE

Modernization enables wider AI adoption

01

Trusted work

Reviewed scenarios, expected outcomes and ownership

AI drafts with domain review

A reviewer can identify a wrong answer

02

Repeatable execution

Reliable suites, isolated data and controlled dependencies

AI drafts tests that run in CI

Another engineer reproduces the result

03

Shared QE services

Reusable environments, contracts, artifacts and support

Assistance works across teams

A second team onboards and operates it

04

Broader AI execution

Evaluated tools, bounded actions and recovery

Agents execute within approved scope

Failures stop, evidence persists, people can intervene

Proposed capability progression. Reviewed assistance can begin while foundations improve. Broader execution requires evidence for its specific scope. Modernization workstreams and sources

For Our Banking Client, fund repeatable payment tests and shared QE services alongside reviewed AI assistance. Validate this capability roadmap through client discovery.

Peer evidence for the payments pilot

Libra Internet Bank

35% less time

Test-creation time

Index: previous time = 100

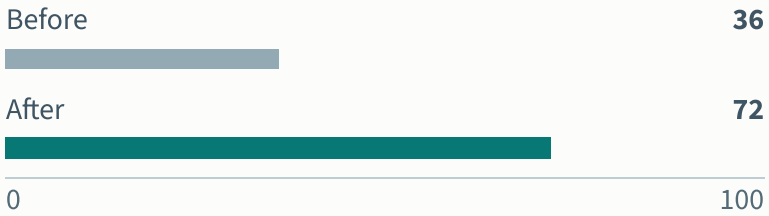
UiPath customer case. Normalized from the reported reduction; review effort is unspecified. F1

Goldman Sachs

36% → 72%

Unit-test coverage

Share of a selected module (%)



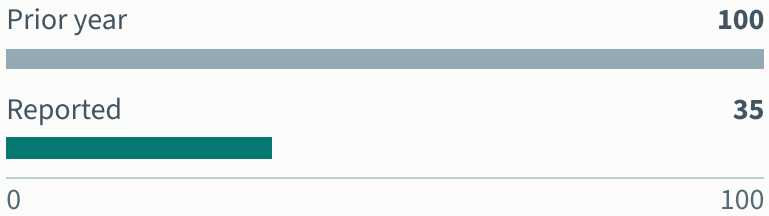
Diffblue customer case. Coverage measures exercised code; it does not establish financial correctness. F2

Fiserv

65% fewer incidents

Major incidents

Index: previous year = 100



Tricentis modernization case. AI testing was a later pilot; this is not an AI-attributed reduction. F4

Separate outcomes and denominators. Each panel has its own meaning; no combined savings rate is calculated.

Peer results support a trial. Outcomes for Our Banking Client remain unmeasured.

Our Banking Client: where to begin

Requirements ready

Onboarding design

Draft scenarios in the current test-management workflow.

Proof

Accepted coverage and measured review effort.

Execution repeatable

Payment API tests

Generate candidates in the existing framework and CI pipeline.

Proof

Correct behavior passes; a known fault fails.

Diagnostics available

Failure triage

Draft diagnoses alongside the existing disposition process.

Proof

Faster correct decisions with traceable evidence.

Proposed entry points. Choose one using the client's constraints; inspect prerequisites and prepare a trial brief.

Choose one entry point, validate its prerequisites and agree how reviewers will judge the result. Wider rollout depends on local evidence.

Observed time framework for the API pilot

Record actual pilot time from baseline through repeated API runs. Example windows remain subject to readiness.

Observation status: not yet recorded

Planning window · weeks 1–2

Observe the baseline

Record API test preparation, active effort, waits and rework for one comparable pack.

Evidence gate: Reviewed API scope + usable baseline

Planning window · weeks 3–8

Run the API pilot

Record setup separately; measure review, execution, triage and retest over repeated packs.

Evidence gate: Required assertions pass + comparable run evidence

Planning window · weeks 9–12

Confirm repeatability

Measure second-team setup, support effort and repeat runs before choosing the next scope.

Evidence gate: Reusable pattern + evidence-based next step

Record actual dates and elapsed time; track active person-hours, waiting reasons, review and rework separately.

Planning windows are illustrative. Environment, data, virtualization and reviewer readiness determine when each phase can progress.

Brainstorm questions

The next decision: assess one payment workflow with QA, development and platform leads.

1. “Across the 75 QA staff, where do people spend time and where are they waiting?”
2. “Which payment journey has stable requirements, observable outcomes and an owner who can join a pilot?”
3. “Could one platform team remove duplicated work across the squads?”
4. “If we return capacity, which backlog or release commitment will use it?”

Agree the bounded pilot and its evidence before promising wider rollout. See the discovery brief for owners, commercial scope and timing.