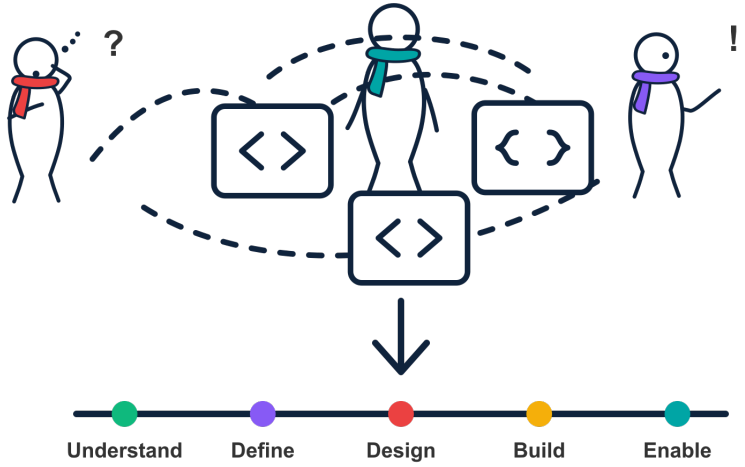
APIOps Cycles

Capability Productization Cycle

From a reusable service promise
to operation and improvement

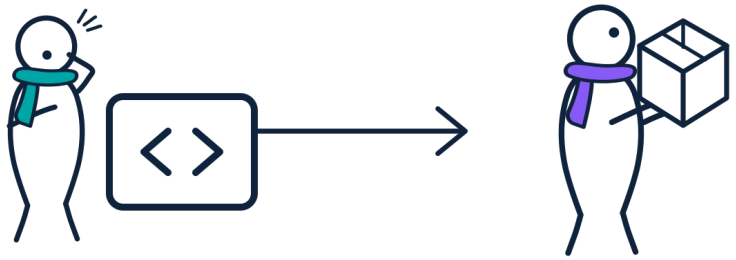


A practical introduction · 25 minutes

Start with the outcome people need

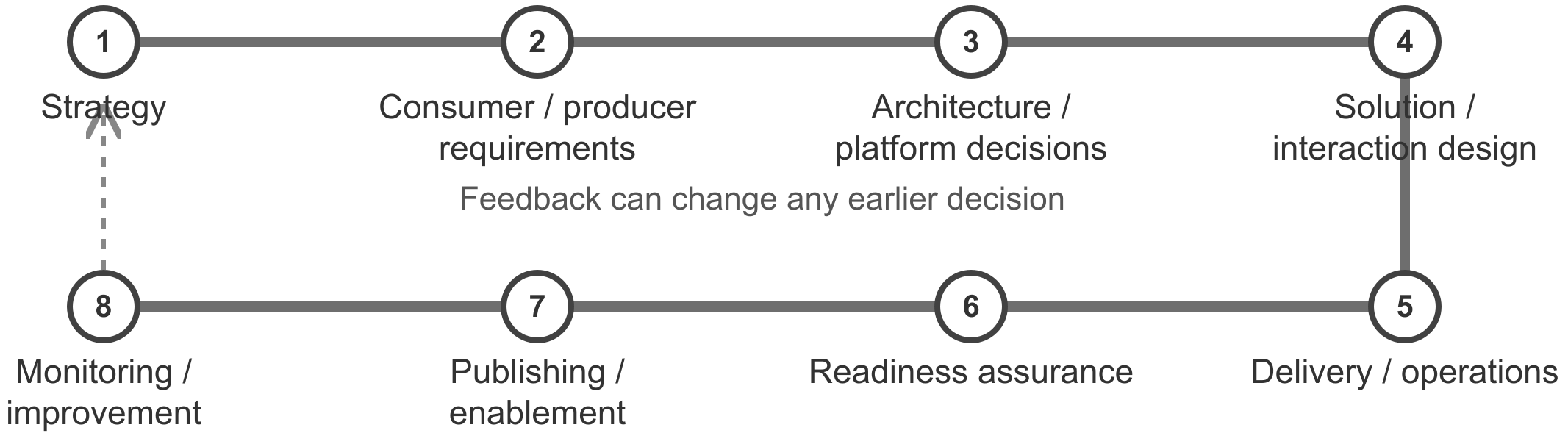
A capability is a reusable ability offered to consumers, with an owner and a service promise.

- Name the consumer and the job they need to complete.
- Define what the producer provides and supports.
- Choose the implementation after testing the need.



Example: let suppliers and service teams understand payment progress without repeated status enquiries.

Eight stations, revisited as evidence changes



Use the stations to expose missing decisions and evidence. Return to earlier work when assumptions change.

1. Strategy: test one capability hypothesis

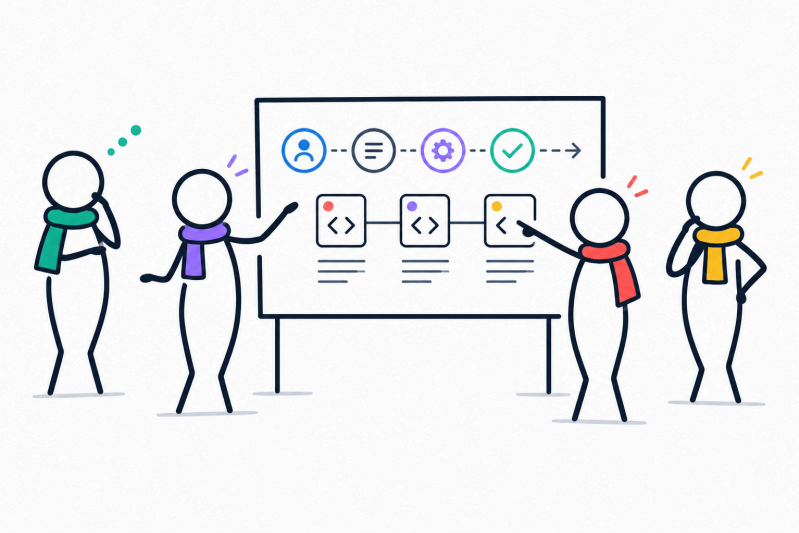
Question	Illustrative working answer
Consumer and job	Suppliers and service teams need to see payment progress.
Problem to validate	Status is scattered; people contact support to explain delays.
Reusable service promise	Provide an authorised, current status and a clear next step.
Value and viability evidence	Observe the journey, review enquiry reasons and test demand.
Decision before investing	Proceed, revise or stop; name the capability owner.

A reusable capability needs evidence of demand and an owner.

2. Requirements: agree the service together

Consumer and producer commitments

- Consumer: describe use, access needs and change constraints.
- Producer: define freshness, availability and support.
- Owner: agree funding, scope and lifecycle decisions.
- Together: test onboarding and the exception path.



Example agreement: show the source timestamp; restrict access to permitted suppliers; explain unavailable status.

3. Architecture: choose how to provide it

Option	Useful when	Evidence to check
API on demand	A consumer needs current status now.	Authorisation, source latency, peak demand.
Event notification	Consumers react when status changes.	Delivery guarantees, ordering, replay.
Reusable integration	Status spans several providers.	Ownership, mapping, reconciliation.
Workflow or automation	Exceptions require coordinated action.	Human approval, fallback, recovery.

Record the choice, alternatives, trust boundaries and operating cost.

4. Design: make the service promise testable

Decision	Supplier-payment status example
Shared meaning	Define submitted, processing, paid and exception states.
Interaction	Read current status; optionally subscribe to changes.
Access and privacy	Identify caller and supplier; return only permitted records.
Unhappy path	Distinguish unknown payment, delayed source and denied access.
Change and compatibility	Specify timestamps, version policy and consumer migration.

Review the interaction with a real consumer before implementation.

5. Delivery includes operation and support

1. Build one complete consumer path from access to useful status.
2. Test permissions, stale data, source failure and recovery.
3. Release with monitoring, support and rollback ownership.
4. Practise the incident and change process with the producer.

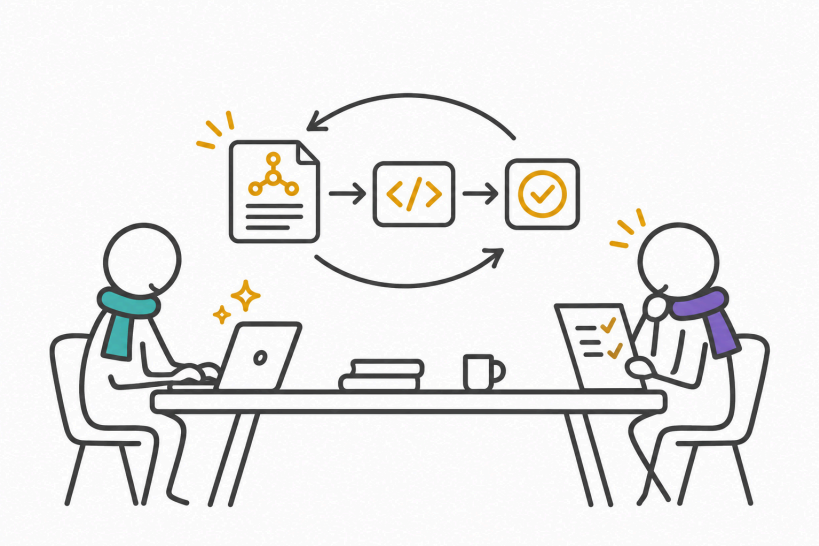
Evidence to keep

- Test results and release record
- Runbook and escalation route
- Dependency and access inventory
- Recovery and lifecycle decisions

Use AI with agreed capability context

Give AI the service promise, boundaries and reviewed interaction—not just a feature request.

- Supply approved domain terms, contracts and acceptance examples.
- Generate one small implementation or test change at a time.
- Review permissions, failures and recovery with a human owner.
- Run the applicable tests, linting and readiness checks.



6. Readiness: decide from evidence

Readiness question	Evidence for the release decision
Does it serve the consumer?	Reviewed service definition and consumer acceptance.
Is access appropriate?	Permission tests and data-handling decisions.
Can it run and recover?	Failure tests, runbook, monitoring and fallback.
Can people use and support it?	Onboarding rehearsal, support route and named owners.
What risk remains?	Recorded exceptions, accountable owner and review date.

Release, release with accepted conditions, or return for more evidence.

7. Publish a capability people can adopt

Catalogue entry	Illustrative content
Purpose and audience	Payment progress for authorised suppliers and service teams.
Access and onboarding	Eligibility, request route, example interaction and first-use test.
Service expectations	Data freshness, supported usage and known limitations.
Ownership and support	Capability owner, producer contact and escalation route.
Lifecycle	Version, change notice, migration and retirement policy.

Publishing includes enablement and support, alongside technical access.

8. Measure value and improve the service

Signal	What to observe	Decision it can inform
Consumer outcome	Enquiry reasons and time to understand a delay.	Is the original problem being reduced?
Adoption and reuse	Active intended consumers and repeat use.	Improve onboarding or revisit demand.
Service health	Status age, failures and recovery time.	Improve source access or resilience.
Cost and lifecycle	Cost to serve, support effort, dependencies.	Invest, change scope or plan retirement.

Agree a baseline and review cadence; take findings back to earlier stations.

Choose a cycle view for the work ahead

Cycle view	How it contributes to the same capability
Capability Productization	Validate value, commitments, readiness and ongoing ownership.
API Productization	Design and operate APIs selected as consumer interfaces.
Integration Productization	Productize reusable connections between providers and consumers.
Automation	Design workflows, human controls, agents and exception handling.

Keep one service promise and owner across implementation choices.

Use canvases to make decisions visible

Decision to work through	Capability Cycle resources
Value and viability	Customer Journey; Capability Value Proposition; Business Model; Hypothesis.
Service commitments	Capability Consumer/Producer Requirements.
Architecture	Business Impact; Locations; Capacity; Capability Architecture Decision.
Solution and interaction	Domain; Capability Solution Design; Interaction.
Operation and readiness	Capability Readiness & Operations; readiness checklist.

Start with the unanswered decision; use its relevant canvas or guide.

Start with one capability and its consumers

1. Choose a consumer journey and write one capability hypothesis.
2. Agree the service promise with the owner, producer and consumer.
3. Compare implementation options and record the first decision.
4. Test a small service interaction and review readiness evidence.
5. Agree what to measure and when to revisit the hypothesis.

Workshop output: an evidence backlog, named decision owners and a next review date.

Make the next reuse easier

A capability is ready for reuse when its value, service promise, access and operational ownership are clear.

1. Name one capability and the consumer outcome.
2. Find the weakest evidence across the eight stations.
3. Assign the next experiment and review the result.

Explore the cycle and its resources
apiopscycles.com/cycles/capability-productization-cycle

APIOps Cycles · Osaango Ltd · CC-BY-SA 4.0
Fictional supplier-payment example; no client results claimed.