# Enterprise AI for Customer Leaders: From Architecture to Applications

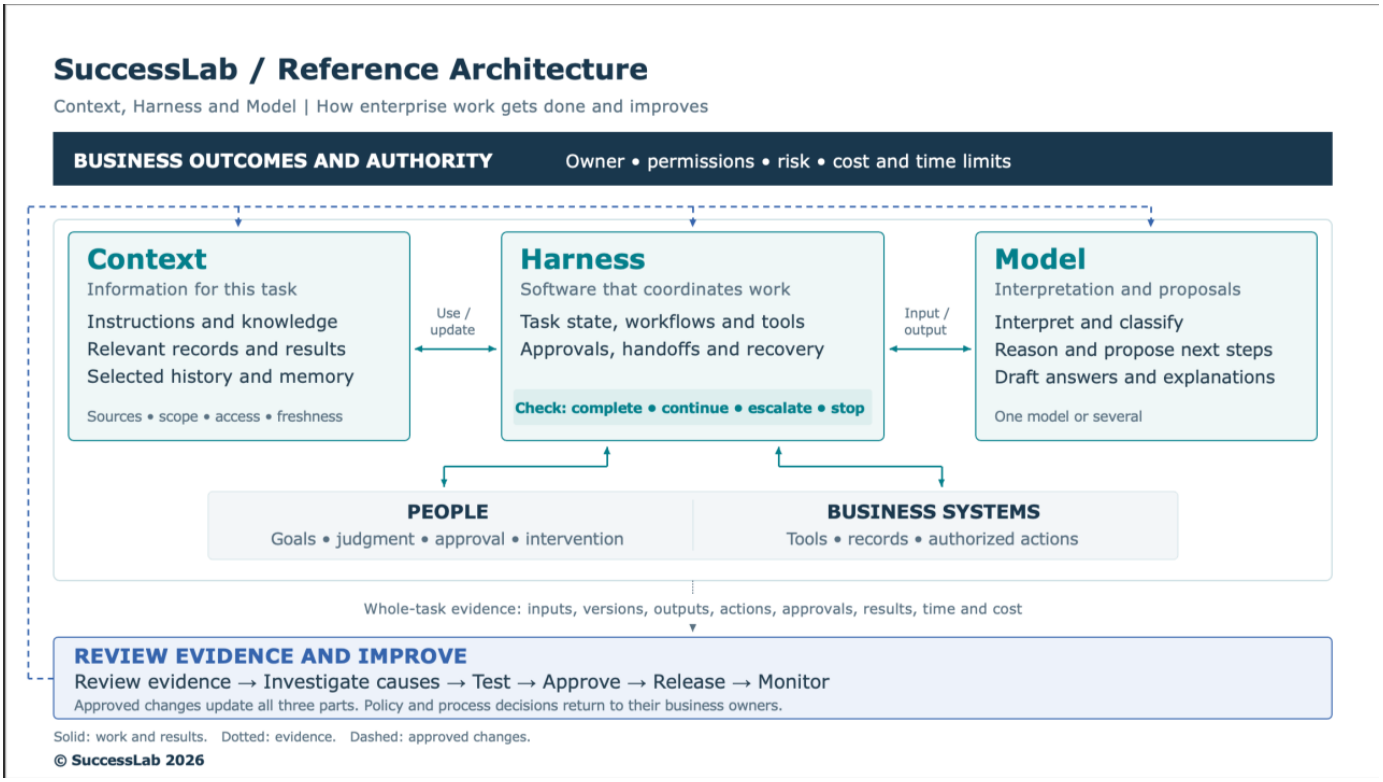
How Context, Harness, and Model work together across service, support, and success.

Omid Razavi · October 19, 2026

## PART I / HOW ENTERPRISE AI WORKS

When an AI system receives a business request, it needs information, a way to coordinate work, and a model to interpret the situation. The SuccessLab Reference Architecture organizes these responsibilities around Context, Harness, and Model.

This guide covers enterprise work that uses language models, workflows and agents. It explains how a task moves through the system and how the organization improves future work. Infrastructure and model training sit outside this view.




Context, Harness and Model: the SuccessLab reference architecture.

The three parts describe different responsibilities. Context is information. Harness is coordinating software. Model is a trained computational capability. One product may supply several parts, or several providers may work together.

## Start with the result and the authority

The top band defines the outcome, its accountable owner, and the limits of the work. It includes permissions, spending limits, deadlines, and decisions that require approval. These requirements apply across the architecture.

For a purchasing task, the result might be an approved replacement component delivered before production stops. Someone must decide which suppliers are acceptable, how much can be spent, and what evidence will confirm delivery.

## Context: the information available for this task

Context is the information prepared for a model at a particular step. It can include the request, instructions, business records, relevant knowledge, previous actions and results from tools. The harness selects permitted information from the available sources. It can retrieve more as the task develops.

Three kinds of information need clear boundaries. Authoritative records establish facts such as an approved contract. Working state records what is happening now, such as an outstanding approval. Memory retains selected information for future use, such as a confirmed preference or a verified lesson.

A company needs rules for resolving conflicts among these sources. In this design, an old conversation summary cannot silently override a current agreement. Retained information needs a source, an appropriate scope, and a way to correct or remove it.

The Context box represents the information. The software that retrieves and manages it belongs in the harness or connected services. Domain owners remain responsible for its meaning and quality. A retrieved document supplies content; permission controls determine what the system may do.

## Harness: the software that coordinates the work

The harness surrounds the model and manages the task. It prepares context, calls models, coordinates tools and people, and tracks progress. It is often called the runtime.

Task state records what happened, what remains open, and who owns the next step. Saving that state, together with action records and recovery rules, helps work resume safely after a delay or failure.

A tool is a callable capability, such as searching records, running a diagnostic, or placing an order. An API is the interface through which software requests that capability. The harness handles requests and responses. The underlying service checks its own permissions and transaction rules.

The harness also coordinates approvals, retries, handoffs, time limits, and escalation. Access controls in the relevant services enforce the boundaries. This gives a request several checkpoints before it can change a business record.

A workflow follows established steps. An agent uses a model to select some of its next steps as information arrives. The harness can coordinate both. A fixed spending rule can sit beside an agent investigating an unusual supply problem.

© SuccessLab 2026 | Omid Razavi

## Model: the capability that interprets and proposes

A model is a trained computational system that produces outputs from inputs. Language models can classify a request, interpret information, prepare an explanation, or propose a next action. The Model box may represent one or more models.

A model can misunderstand the task or produce an incorrect answer. Outputs can vary between attempts. An early mistake can also affect later steps. A fluent explanation therefore needs the same scrutiny as any other proposed result.

An agent combines a model with software that lets it pursue a goal through several steps. Its performance depends on the combination of context, harness, and model. Test that combination when changing any of the three.

## Follow one task through the architecture

Consider an illustrative purchasing task. A shipment is late. The harness opens the task and gathers the component specification, approved suppliers, inventory, budget, and delivery deadline. It asks a model to assess the choices.

The model requests a stock check. The harness calls the supplier tool and adds the returned information to the next context. The exchange continues until it produces a workable proposal or requires human help.

A manager reviews any decision requiring approval. The purchasing service executes the authorized order. Its retry rules must prevent a delayed response from creating a duplicate purchase.

The harness coordinates the checks. An order number confirms that an order exists. Delivery confirmation establishes the later milestone. An outstanding delivery stays open with a next action and an owner. This repeated exchange is the execution loop.

People contribute goals, expertise, approval and intervention through conversations, forms or specialist workspaces. Business systems provide records and execute transactions. Tasks can also begin with an event, such as the shipment alert, before anyone opens an application.

## Check this result and test the system

Verification checks whether the current task met its completion conditions. It might inspect a transaction, test a report, or obtain confirmation from a person. The harness then completes the work, continues, pauses for help, or stops.

Evaluation tests how reliably the system performs across a set of cases. It should include ordinary work, difficult exceptions, and known failures. Checks against records and rules can establish some results. Expert review helps assess judgment and communication. Model-based grading also needs calibration against trusted checks or human assessment.

© SuccessLab 2026 | Omid Razavi

The dotted path in the diagram represents evidence from the whole task: relevant inputs, component versions, model outputs, actions, approvals, results, time and cost. Access and retention rules apply to this record. It supports investigation without assuming that every internal model step is visible.

### Improve the next round of work

The improvement loop reviews evidence, investigates causes, tests changes, approves release, and monitors what happens afterward. A Context change might remove outdated guidance. A Harness change might repair a handoff. A Model change might select a better-suited capability.

Some findings require a business decision. Repeated purchasing delays may justify reviewing an approval threshold. The process owner decides whether to change it. Technical teams supply evidence and implement the authorized choice.

Test a proposed change against the failure and against work that already performs well. Then monitor the released version. Restoring an earlier configuration can correct future behavior, while actions already taken may need separate recovery.

Use the three parts as a starting point for investigation. Was the right information available? Were the steps and permissions handled correctly? Did the model perform its part well? Several causes may contribute. Follow the evidence before assigning the fix.

Note: The Context, Harness, and Model framing draws on Harrison Chase of LangChain. The enterprise explanation and examples are a synthesis from SuccessLab.

## PART II / CUSTOMER APPLICATIONS

In Part 1, I described the SuccessLab Reference Architecture: Context supplies information, the Harness coordinates work, and the Model interprets and proposes. The architecture connects these responsibilities to people, tools, and business systems, with verification and an improvement loop.

Here, I want to show how those pieces become applications for customer organizations.

Consider three kinds of work: resolving a billing dispute, recovering from a technical failure, and helping a customer improve a business process. Each requires a different combination of information, model assistance, system actions, and human decisions.

The application takes shape when we configure those capabilities around a defined kind of customer work. Each refund dispute, support incident, or improvement plan becomes an individual case, with its own history, status, permissions, and outstanding commitments.

### What makes the architecture an application?

An application needs a defined scope: what starts the work, what it is authorized to do, and what will establish completion.

For cancellation disputes, that means identifying the relevant evidence, applying policy, obtaining any required approval, executing corrections, and checking the result. Those choices determine which records the harness retrieves, what the model is asked to interpret, which tools are available, and when a person must intervene.

The application is the configured combination of those pieces. It can handle many cases of the same kind, including exceptions that require different paths.

Context changes as each case develops. The harness may first prepare the customer's request and account records for the model, then add a payment result or an authorized decision at a later step. The model's proposal becomes useful when there is a permitted route to act on it and a way to check what happened.

Tools provide capabilities such as retrieving records, running diagnostics, or requesting a transaction. The underlying services enforce their own permissions and transaction rules. People contribute expertise, approvals, and decisions within their authority.

These capabilities can come from several products and services. The application design specifies how they cooperate around the work.

## Give the work somewhere to continue

People need ways to participate in the application. A customer may begin through a conversation. An engineer may join through a support workspace. A billing event may start a case before anyone opens a screen.

The harness connects those interactions to the case and maintains its working state. An approval may arrive tomorrow. Engineering may need another release. The customer may need time to change its process. Work must resume with an accurate account of what happened and what remains open.

Some steps follow established workflows. Others require investigation, with a model helping select the next check as evidence arrives. An application can combine both, with different levels of discretion at different steps.

A named person owns follow-through within their authority and escalates decisions that belong elsewhere. The application keeps those dependencies and commitments visible.

The following examples show how this construct supports three different kinds of customer work.

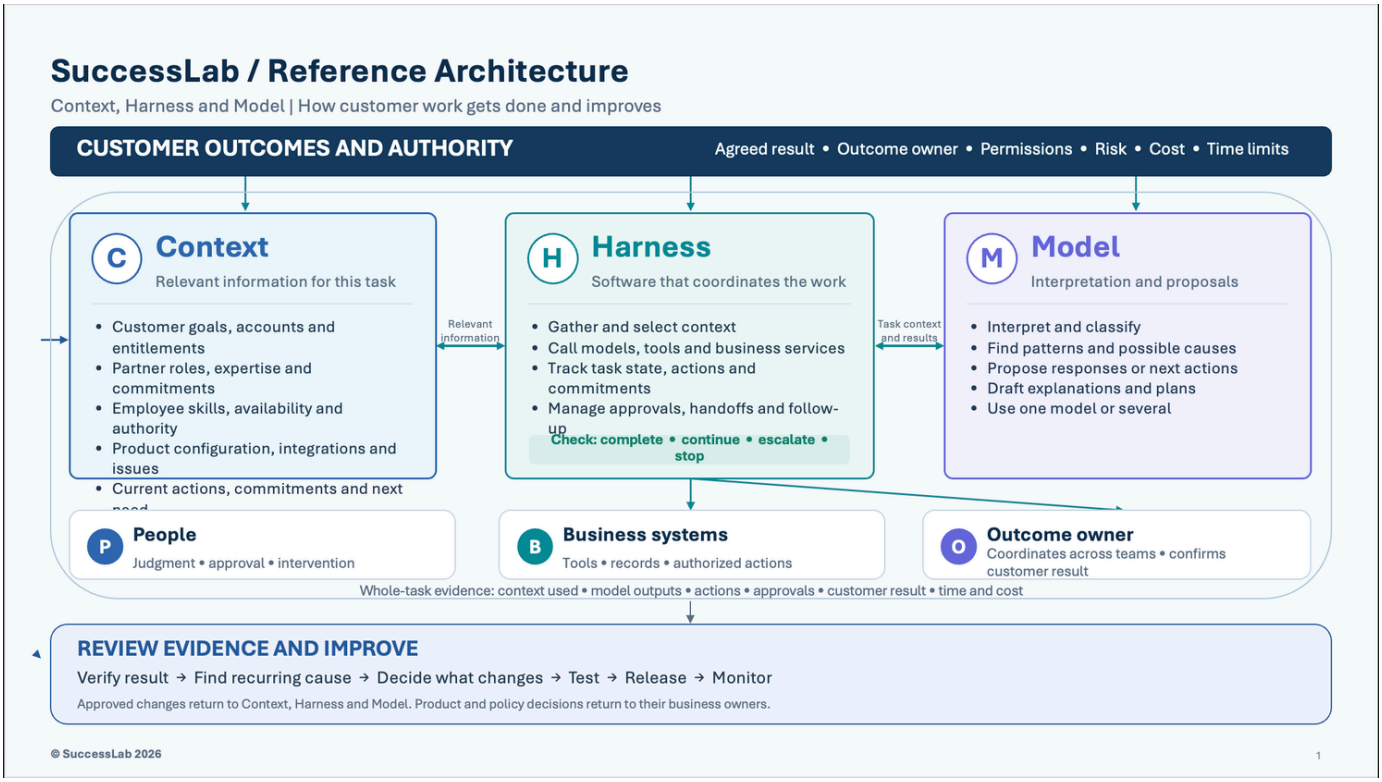



Figure 1. SuccessLab Reference Architecture. Context, Harness and Model connect to people and business systems. The outcome owner follows the customer result across teams.

## B2C customer service: the customer canceled, but was charged again

A subscriber receives another annual charge after a service representative confirmed their cancellation. They want a refund.

Context brings together the cancellation request and payment record, including the earlier assurance. The model helps reconstruct what happened. The harness routes a correction within approved limits or sends an exception to an authorized representative. Billing executes the approved refund and protects against duplicate adjustments.

The service owner checks that cancellation is recorded and follows the refund through settlement. The subscriber gets a clear update without having to contact another team.

I would then ask how we confirmed a cancellation that billing had not recorded. Give that failure an owner. A check before sending the confirmation could prevent another customer from having the same experience.

## B2B customer support: orders stop reaching the platform

A connector update interrupts order synchronization during a peak sales period. The customer has a growing backlog and little room for disruption.

The support team needs the affected configuration, error logs, previous troubleshooting, and business impact. The model suggests useful checks. The harness coordinates permitted diagnostics, brings engineering into the investigation, and preserves the history.

Now comes the judgment call. A tested workaround may restore processing sooner than a permanent patch. Support and engineering assess the risks with the customer, who authorizes changes in its environment. The relevant systems execute the approved action. Replaying failed orders requires controls against duplication.

The incident stays open until new orders flow, and the backlog is reconciled. If a workaround restored service, the permanent correction remains a separate commitment. I would also ask which other customers have the affected configuration and whether a targeted check could prevent their next incident.

## B2B customer success: CRM activity is healthy, but deals still stall

A customer bought a CRM platform to shorten sales cycles. Salespeople update opportunities regularly, but quotes and discount approvals still move through email. Renewal is approaching, and the original sponsor has left.

Context should include the business case and opportunity history, along with open issues and past commitments. The model can help locate delays and prepare questions. The customer success manager reviews the findings with sales operations. They confirm that missing pricing information and unclear ownership of approvals are delaying proposals.

The harness tracks the agreed response. Services repairs the integration. Sales and finance leaders settle approval thresholds. Customer sales operations owns the revised process, while the CSM follows through on the provider's commitments.

First, check whether quotes and approvals move faster. Then review sales cycles for comparable opportunities, allowing for differences in deal size and complexity.

Before discussing more licenses, I would sit down with the new sponsor and review the value delivered. What has improved? What do we still owe them? That conversation should shape the expansion proposal.

© SuccessLab 2026 | Omid Razavi

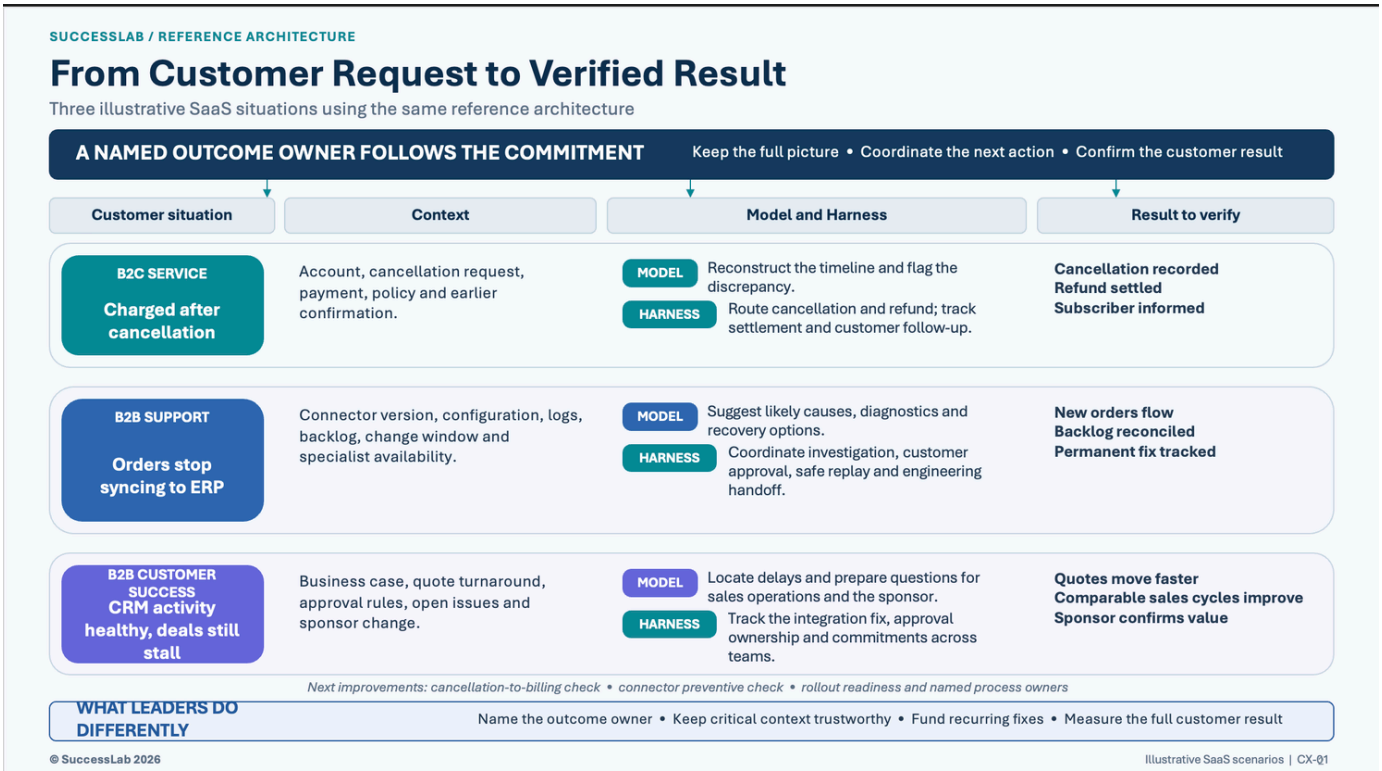



Figure 2. From customer request to verified result. Each example has a completion test and an owner for unfinished work.

## One architecture, three kinds of work

The examples share an architecture, but the applications need different ways to proceed and establish completion.

The service application resolves a bounded transaction. The policy, approval limits, and required corrections are relatively well defined. The application must reconcile what the customer was promised with what the systems recorded, then follow the correction through settlement.

The support application manages an investigation and recovery. Its path depends on what the team discovers. It needs to carry evidence between steps, accommodate specialist judgment, and coordinate actions whose sequence matters. Restoring service and delivering a permanent correction can have separate completion conditions.

The success application supports a longer effort to realize value. It can organize evidence, prepare analysis, track agreed interventions, and prompt reviews. The CSM remains responsible for the provider's follow-through. Customer leaders retain authority over their process and decisions.

That third example requires particular care. Completing an integration repair establishes that a piece of work was delivered. Faster quotes and approvals provide evidence of operational improvement. Assessing the contribution to shorter sales cycles requires a broader review with the customer.

The application needs to represent those milestones separately. Otherwise, a completed task list can create unwarranted confidence that the business objective has been achieved.

Across all three applications, the design choices are concrete: how much discretion the model has, which actions are available, when a person must decide, and what evidence lets the work advance.

## Make the application assessable

A useful application description should let us follow a case from its trigger through decisions, actions, handoffs, and completion checks. At each stage, we should be able to see what's established, what remains uncertain, and who owns the next commitment.

Evaluation then tests that configured application across cases. Can it handle conflicting records, delayed approvals, failed actions, and interruptions? Can it recognize when the available evidence is insufficient and obtain help?

Evidence from those cases supports deliberate improvements. A team may correct the context, repair a handoff in the harness, change model instructions, or revisit a business rule with its owner. The revised combination needs testing before release.

For customer leaders, I would begin with one recurring problem and ask the team to demonstrate the entire application - including an exception and a pause for human judgment.

That demonstration makes the architecture tangible. It shows how the pieces cooperate, how unfinished work stays visible, and how we determine whether the customer commitment has been fulfilled.

© SuccessLab 2026 | Omid Razavi

# Appendix: Customer application flows

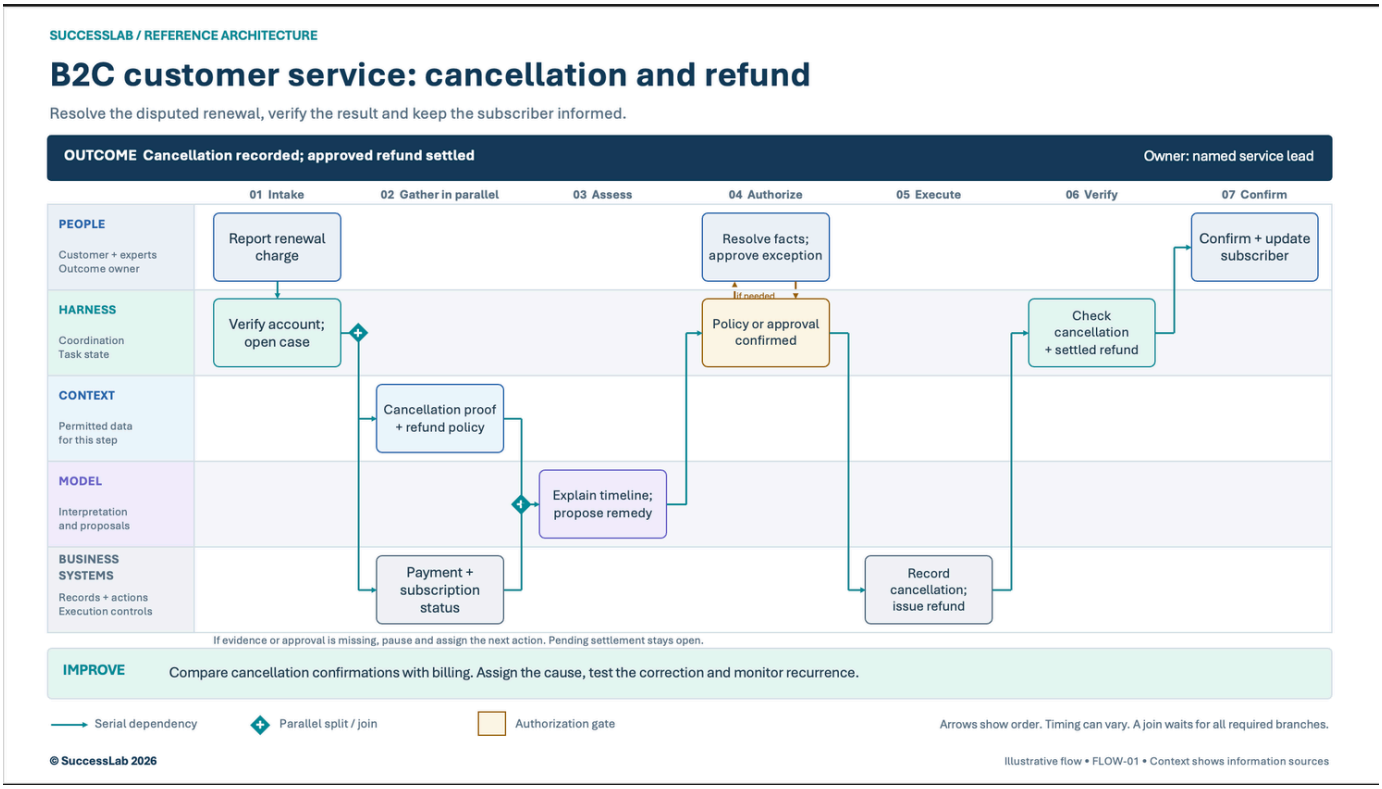



Figure 3. Cancellation evidence and current billing records can be gathered in parallel after account verification. Assessment, authorization, execution, and verification follow in sequence. Pending settlement remains an owned commitment.

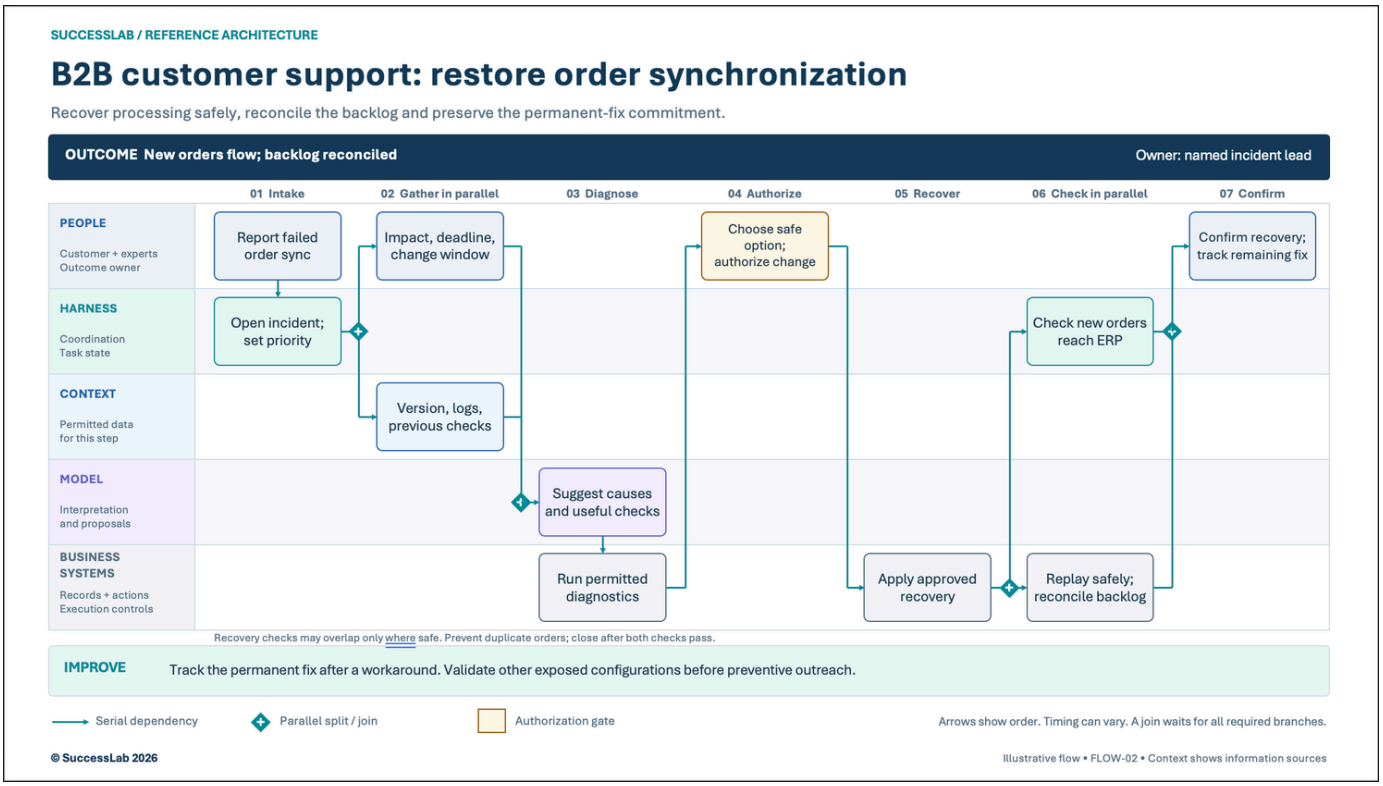



Figure 4. Assess impact and gather evidence in parallel. After customer-authorized recovery steps, live-order checks and backlog replay may overlap where sequencing and duplicate controls permit. Both must pass to confirm recovery.

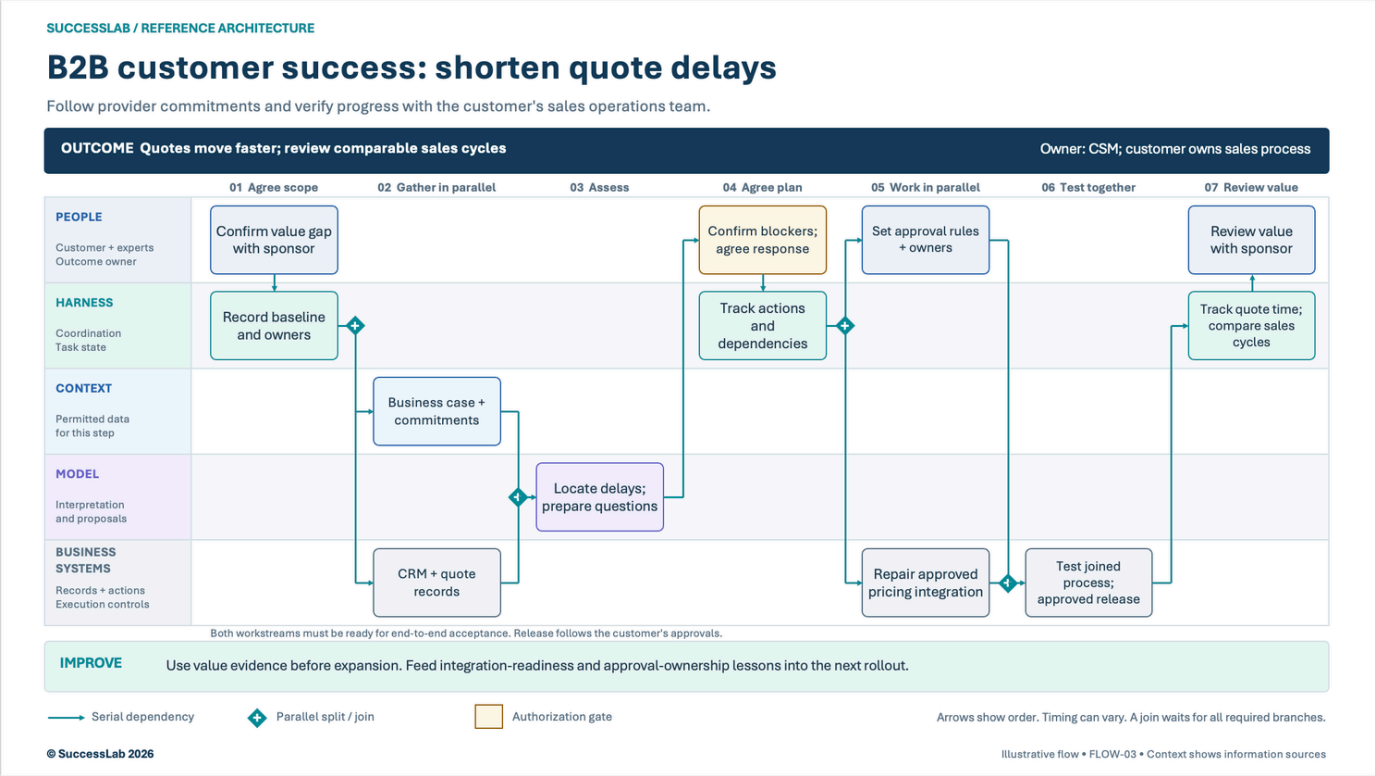



Figure 5. After scope agreement, pricing-integration repairs and approval-rule decisions run in parallel. Complete both before end-to-end acceptance and approved release. The CSM owns follow-through; the customer owns its sales process and results.

Original LinkedIn sources: 1 · 2