

logiCells

Application Modernization Guide

Recovering, Structuring and Operationalizing
Business Knowledge

From legacy behavior to explicit,
validated business knowledge

Editorial Note

Application modernization is not simply the replacement of software. It is the recovery of business decisions, distinctions and obligations that an organization has gradually embedded in technology and operational practice. This guide proposes a disciplined method for making that knowledge explicit and using it as an independently governed asset.

The running order-management case is fictional. All examples are technology-neutral and illustrate a possible target specification, not claims about an executable product or an automatic extraction capability. The purpose is to make the method concrete enough to be challenged, applied and tested.

Two questions drive the book: *What constitutes structured business knowledge? How can the business knowledge embedded in an existing application be made visible, justified and useful in a modernization program?*

Scope. Methodology and illustrative examples; this is not a product implementation manual.

ENGLISH EDITION 1.0

Based on *logiCells Application Modernization Guide*, edition 1.0, October 2026. The examples and technical caveats of the source are preserved.

Contents

Editorial Note	i
1 Modernization Begins with Meaning	1
1.1 Three layers of an existing application	1
1.2 Why a code rewrite is insufficient	1
1.3 The modernization contract	1
2 What Structured Business Knowledge Means	2
2.1 From labels to governed definitions	2
2.2 Separate facts, rules and decisions	2
2.3 Formal does not imply correct	2
3 Discovering the Existing Application	3
3.1 Build an evidence map	3
3.2 An evidence ladder	3
3.3 Practical discovery instruments	3
4 A Running Case: Confirmation of a Sales Order	4
4.1 The legacy fragment	4
4.2 The decisive gap: present versus valid	4
4.3 Register BR-ORD-014	4
4.4 Classify rather than copy	5
5 From Hypothesis to Agreed Business Rule	6
5.1 Resolve concrete scenarios	6
5.2 An example of an approved policy	6
5.3 Traceability as the control mechanism	6
6 An Independent Business Knowledge Model	7
6.1 Identity, relations and constraints	7
6.2 A portable specification of BR-ORD-014	7
6.3 Knowledge versus a conventional specification	8
6.4 A lifecycle is a contract	8
6.5 Operationalizing without coupling to a platform	8
7 Proving the Model with Tests	9
7.1 Acceptance cases derived from the specification	9
7.2 Characterization versus acceptance	9
7.3 Differential validation	9
7.4 Release gate	10
8 Incremental Replacement and Coexistence	11
8.1 Choose a narrow business seam	11
8.2 Three rollout stages	11
8.3 State ownership and transactions	11
9 Data Reliability and Semantic Migration	12
9.1 Correct data can still have the wrong meaning	12
9.2 Reconciliation on two axes	12
9.3 A tangible data-to-knowledge interface	12
10 AI Supported by Explicit Business Knowledge	13

10.1 From retrieval to controlled interpretation

10.2 An order confirmation inquiry

10.3 Measure groundedness, not readiness slogans

11 A Practical Modernization Pilot14

11.1 Team and roles

11.2 Indicative sequence

11.3 When not to use this approach

11.4 Pilot scorecard

12 Governance and Continuous Evolution15

12.1 Keep knowledge independent of one codebase

12.2 Avoid a second undocumented legacy

12.3 The distinctive outcome

A Business Knowledge Record Template16

B BR-ORD-014 Traceability Matrix17

C Applying the Method with logiCells18

C.1 The role of logiCells

C.2 A practical engagement boundary

C.3 Contact and next step

C.4 What should be demonstrated before adoption

D Readiness Checklist19

Modernization Begins with Meaning

1.1 Three layers of an existing application

A business application contains an *implementation layer* (code, storage, interfaces, infrastructure), an *observational layer* (transactions, logs, incidents and actual outcomes), and a *semantic layer* (concepts, obligations, decisions and accepted meanings). The layers are related but not interchangeable. A database may reliably record a status without explaining what commitment that status creates.

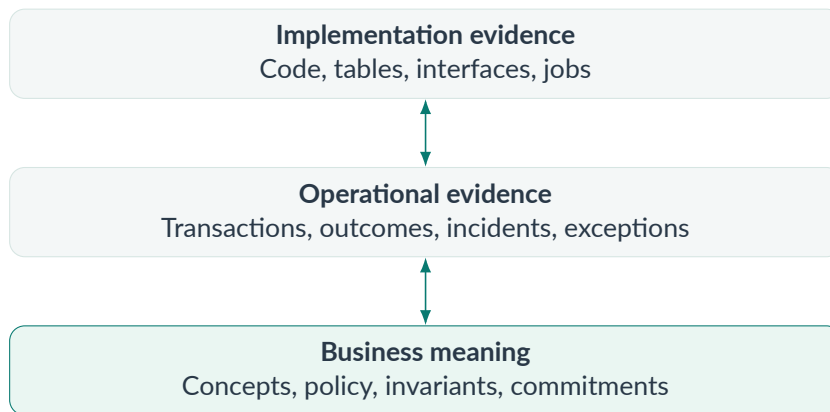


Figure 1.1: Three layers to investigate; none alone establishes business intent.

1.2 Why a code rewrite is insufficient

Recreating implementation details can preserve obsolete rules, accidental behavior and hidden contradictions. Conversely, redesigning without evidence can eliminate requirements no one knew to state. The goal is not identical code; it is justified continuity of business meaning, with approved changes made explicit.

Principle. Treat the legacy system as a source of evidence, not as the unquestioned definition of the business.

1.3 The modernization contract

Agree at the outset on critical decisions to preserve, known defects to correct, owners empowered to adjudicate ambiguity, and evidence required before switching traffic. A successful outcome consists of an explicit knowledge baseline, a tested replacement capability and a controlled path to operational adoption.

Deliverable: Modernization charter linking business outcomes, semantic risks and acceptance criteria.

What Structured Business Knowledge Means

2.1 From labels to governed definitions

Structured knowledge is not a glossary alone. It gives identity to concepts, defines their relationships, constrains valid states, identifies decisions and connects rules to their provenance and tests. It is structured when independent participants can review a claim, detect an inconsistency and decide whether two implementations honor it.

Element	Illustration in order management
Concept	A sales order is a request that can become an accepted commercial commitment.
Relationship	A sales order belongs to a customer and contains order lines.
Invariant	A confirmed order must satisfy the approved confirmation eligibility policy.
Transition	Confirmation changes a draft into a confirmed commitment.
Decision	Is this specific order eligible for confirmation now?
Evidence	Code path, transaction trace, approved policy and acceptance test.

Table 2.1: Elements of structured business knowledge.

2.2 Separate facts, rules and decisions

A *fact* reports an observed or recorded condition. A *rule* defines what must or may hold. A *decision* applies relevant rules to facts in a specified context. These distinctions matter for AI: a generated explanation may refer to facts without being authorized to change the rules or trigger a business action.

2.3 Formal does not imply correct

A diagram or a rule expression makes assumptions inspectable, not true. The business owner must resolve questions such as whether “valid line” requires an active product, a positive quantity, a permissible price or successful allocation. Record open questions as first-class artifacts rather than guessing an answer.

Deliverable: Domain glossary, policy register, lifecycle map and unresolved-question list.

Discovering the Existing Application

3.1 Build an evidence map

Start with capability boundaries, databases, service calls, batch jobs, operational reports, user roles and exceptional handling. Capture representative transaction traces and deployment configuration. Link each proposed rule to the artifacts that justify it, with version identifiers and observation dates.

3.2 An evidence ladder

1. **Candidate:** inferred by an analyst or AI from an artifact.
2. **Observed:** reproduced in code analysis or runtime evidence.
3. **Corroborated:** supported by independent sources or stakeholders.
4. **Approved:** accepted as intended policy by an accountable business owner.
5. **Verified:** covered by executable acceptance tests against the target.

The ladder tracks different forms of confidence; it does not mean that a test on legacy code alone establishes intended policy. AI may accelerate indexing and hypothesis generation, but cannot promote an unreviewed inference into an approved business rule.

3.3 Practical discovery instruments

Static analysis reveals branches, validation paths, database writes and dependencies. Database profiling reveals data distributions and outliers. Logs, traces and incident histories expose exceptional flows. Process mining may illuminate actual workflow variants where event quality is adequate. Interviews resolve intended meaning and business consequences. Use synthetic or appropriately protected data and record evidence limits.

Deliverable: System map, evidence register and ranked catalogue of candidate rules.

A Running Case: Confirmation of a Sales Order

4.1 The legacy fragment

Consider an illustrative implementation:

```
if Order.Status = Draft then
  if Order.LineCount > 0 then
    if Customer.IsActive then begin
      Order.Status := Confirmed;
      ReserveStock(Order);
    end;
```

Listing 4.1: Illustrative legacy confirmation path

This fragment suggests eligibility checks and a side effect. It does not establish whether reservation failure rolls back confirmation, whether duplicate requests are safe or whether a nonempty order has any usable lines.

4.2 The decisive gap: present versus valid

The code checks `LineCount > 0`. The business statement proposed during discovery is “an order may be confirmed only if it contains at least one valid line.” These conditions differ. An order containing one cancelled line passes the legacy test but may violate the proposed business policy. That discrepancy is not a minor implementation detail: it is precisely the knowledge gap that modernization must surface.

Principle. Never silently replace observed legacy behavior with a stronger business rule. Record the discrepancy, obtain a decision, and test both behaviors.

4.3 Register BR-ORD-014

Field	Reviewed working entry
Identifier	BR-ORD-014
Business question	When may a draft sales order become a confirmed commitment?
Legacy observation	A draft order passes if it has more than zero lines and its customer is active.
Proposed target rule	A draft order may be confirmed if the customer is eligible and at least one line satisfies the approved line-validity policy.
Open definitions	Eligible customer; valid line; excluded-line treatment; reservation atomicity; duplicate submissions.
Owner	Order-management policy owner (to be assigned).

Field	Reviewed working entry (continued)
Evidence level	Observed for legacy condition; Candidate for proposed policy (separate evidence assertions).
Policy status	Draft / pending approval.
Classification	Deprecated legacy behavior (provisional disposition); target correction awaiting business approval.
Evidence	Source path, production-like trace, workshop decision and test identifiers (to be attached).

4.4 Classify rather than copy

For every discovered behavior use one of four dispositions: *required*, *tolerated*, *deprecated* or *defect*. A tolerated behavior might be supported only during migration; a defect should not be reimplemented merely because it is observable. Log every classification with a rationale and approver.

From Hypothesis to Agreed Business Rule

5.1 Resolve concrete scenarios

The domain workshop should answer questions through examples: an empty order; one cancelled line; one active line with zero quantity; an inactive customer; insufficient stock; a second confirmation request; a concurrent cancellation. The workshop outcome is a decision table, not just meeting minutes.

Scenario	Target decision	Reason
Draft; active customer; valid line	Allow	Policy conditions hold
Draft; active customer; only invalid lines	Deny	Valid-line condition fails
Draft; inactive customer; valid line	Deny	Customer ineligible
Already confirmed; repeated request	Explicit policy	Idempotency decision required
Draft; valid line; no stock	Deny	Failed reservation prevents confirmation under illustrative target policy
Draft; one valid and one cancelled line	Explicit policy	Define exclusion, retention and reporting
Draft; partial reservation succeeds	Compensate or atomically undo	No stock should remain reserved after failure

Table 5.1: Order-confirmation decision scenarios.

5.2 An example of an approved policy

For demonstration only, suppose the business approves the following definitions: a valid line has a sellable product, a strictly positive quantity, and a non-cancelled status; customer eligibility requires active status. All lines remain in the order for traceability; only valid lines enter stock reservation, including exclusion of cancelled lines, zero-quantity lines and lines with non-sellable products; this is an explicit illustrative choice requiring business approval.

Reservation is all-or-nothing: any partial stock hold is undone before the order remains **Draft**. A repeated request with the same idempotency key must not reserve twice or emit a second confirmation. Confirmation succeeds only if reservation succeeds. These are illustrative decisions, not facts inferred from the earlier code fragment.

5.3 Traceability as the control mechanism

Maintain links from observation to hypothesis, business decision, model element, implementation and acceptance test. When one link changes, identify impacted cases and stakeholders. Semantic equivalence is an auditable argument supported by this chain, not an assumption based on matching row counts.

Deliverable: Approved decision table, disposition register and traceable policy version.

An Independent Business Knowledge Model

6.1 Identity, relations and constraints

Model `SalesOrder`, `Customer`, `OrderLine` and `Product` as concepts, not as direct copies of database tables. Specify identity and lifecycle independently of physical keys. State relations (the order has lines; each line references a product; the order has a customer) and explicitly document cardinality, ownership and temporal interpretation.

6.2 A portable specification of BR-ORD-014

The following is platform-neutral specification pseudocode. It is not asserted to be a runnable DSL.

```
ValidLine(line) :=
  line.quantity > 0
  AND line.product.isSellable
  AND line.status != Cancelled

CanConfirm(order) :=
  order.state = Draft
  AND order.customer.isActive
  AND EXISTS line IN order.lines: ValidLine(line)

Confirm(order):
  IF completed result exists for request key:
    RETURN previous result without new side effects
  PRECONDITION CanConfirm(order)
  ELIGIBLE_LINES := FILTER order.lines BY ValidLine
  EXCLUDE all invalid lines from reservation
  TRY atomic reservation of ELIGIBLE_LINES:
    IF all reservations committed:
      TRANSITION order Draft -> Confirmed
      EMIT OrderConfirmed once
    ELSE:
      RELEASE any partial reservations
      KEEP state Draft; record failure
```

Listing 6.1: Technology-neutral policy specification

The request key is checked before state-dependent eligibility; otherwise a retry after confirmation would fail the **Draft** precondition. Concurrent requests also require a durable, atomic idempotency record, including safe handling while the first request is in progress. A database transaction, reservation service or compensating saga must enforce the approved all-or-nothing outcome.

A failed reservation is not a rollback of a completed confirmation: the order must never enter **Confirmed** in that case. If some reservations succeed before another fails, compensating release or an atomic reservation protocol is necessary. The ordering of stock reservation, state update and

event publication needs a transaction design. The specification records the intended observable contract while leaving implementation mechanisms open.

6.3 Knowledge versus a conventional specification

A conventional functional specification commonly describes a feature or a screen at a point in time. A governed knowledge model connects business identities, definitions, eligibility conditions, state changes, ownership, evidence, policy versions and executable verification across implementations. A well-written specification can contain all these elements; the distinction is not a document format but whether meaning is explicit, independently maintained, traceable and reusable. In BR-ORD-014, the same definition of **ValidLine** supports confirmation decisions, user explanations, test design and impact analysis when the policy changes.

6.4 A lifecycle is a contract

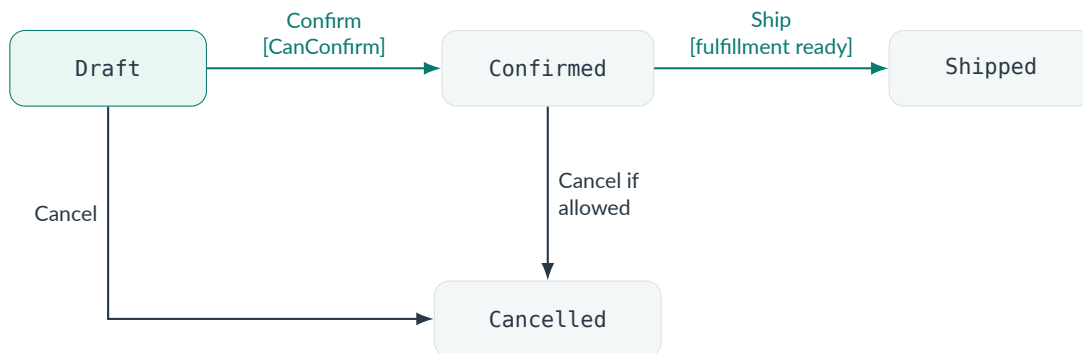


Figure 6.1: Illustrative order lifecycle: guards must be decided and tested.

6.5 Operationalizing without coupling to a platform

The knowledge model becomes useful when it informs multiple controlled artifacts: a decision service, UI eligibility messaging, validation tests, documentation and AI-assisted explanations. These uses must share governed semantics while maintaining their own security and deployment boundaries. A model need not itself be an executable engine to provide value; if execution is required, its implementation must be tested against the model's contract.

Deliverable: Versioned concept model, formalized policy, transition contract and mapping to implementations.

Proving the Model with Tests

7.1 Acceptance cases derived from the specification

```

AC-014-01
GIVEN Draft order, active customer, one valid line
WHEN Confirm is requested
THEN state becomes Confirmed; confirmation event recorded

AC-014-02
GIVEN Draft order, active customer, one cancelled line
WHEN Confirm is requested
THEN confirmation rejected; state remains Draft

AC-014-03
GIVEN Draft order, inactive customer, one valid line
WHEN Confirm is requested
THEN confirmation rejected; state remains Draft

AC-014-04
GIVEN Draft order, active customer, one valid line
AND stock reservation fails
WHEN Confirm is requested
THEN state remains Draft; any partial holds are released
AND failure is auditable; no confirmation event emitted

AC-014-05
GIVEN Draft order, one valid and one cancelled line
WHEN Confirm is requested
THEN reserve only the valid line; retain both lines
AND record exclusion in decision audit

AC-014-06
GIVEN a previously successful confirmation with request key K
WHEN the same confirmation request K is retried
THEN return prior result; reserve no stock again
AND emit no duplicate OrderConfirmed event

```

Listing 7.1: Illustrative specification-level acceptance cases

7.2 Characterization versus acceptance

A characterization test documents what the legacy system does. An acceptance test checks approved target behavior. In the cancelled-line scenario the tests may deliberately disagree; that disagreement is an approved correction, not automatically a regression. Maintain both as separate suites.

7.3 Differential validation

Run comparable cases against old and new paths in isolated environments. Compare decisions, state changes, downstream effects, idempotency and audit records. Classify each difference as a

known improvement, an approved tolerance or an unresolved regression. Matching counts and totals do not by themselves prove that business meaning has been preserved.

7.4 Release gate

Require signed policy decisions for high-impact rules, passing target acceptance tests, reviewed exception handling, release-reversal rehearsals, reservation-compensation tests and monitoring in place. High-risk cases involving money, contracts or regulatory obligations should receive proportionate controls.

Deliverable: Executable case catalogue, differences register and release-approval evidence.

Incremental Replacement and Coexistence

8.1 Choose a narrow business seam

Start with a decision that has clear boundaries, observable outcomes and business owners. For the order case, the eligibility decision may be a better pilot than replacing the entire order-processing module. Existing applications can continue owning data while the new decision path is evaluated.

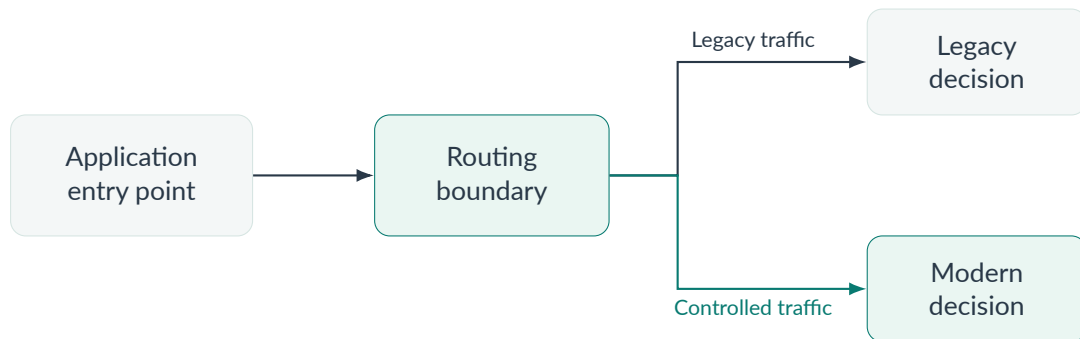


Figure 8.1: A controlled seam enables shadow evaluation and partial routing.

8.2 Three rollout stages

1. **Shadow:** compute the modern decision with no operational effect; collect explained differences.
2. **Partial routing:** direct a small eligible cohort to the new decision under feature controls.
3. **Retirement:** remove the legacy branch after evidence, dependency checks and rollback criteria have been satisfied.

8.3 State ownership and transactions

Assign an authoritative writer for each state and define integration contracts. Avoid dual writes without reconciliation. Make event ordering, retries, duplicate delivery and eventual consistency explicit; eventual consistency becomes a business-policy question when it affects commitments, stock or money.

Deliverable: Cutover plan, routing controls, ownership matrix and rollback procedure.

Data Reliability and Semantic Migration

9.1 Correct data can still have the wrong meaning

Quality controls answer whether facts are complete, consistent, timely and traceable. Semantic controls answer what those facts mean. A status code can be perfectly populated and still conflate “approved”, “reserved” and “committed”. For each migrated field, record the domain interpretation and any transformation assumptions.

9.2 Reconciliation on two axes

Check quantitative reconciliation (records, balances, key coverage) and semantic reconciliation (states, obligations, rule outcomes, temporal meaning). The two are complementary. A successful migration must also maintain lineage and make losses of precision explicit.

9.3 A tangible data-to-knowledge interface

For the order-confirmation case a quality-controlled source might provide `customer.active`, product sellability, line quantities and cancellation status. The business model defines how these facts participate in `CanConfirm`. The decision result should expose a reason code and reference the applied policy version. Reliability makes inputs trustworthy; semantics makes their use intelligible.

Deliverable: Field-to-concept mapping, reconciliation reports and policy-versioned decision contract.

AI Supported by Explicit Business Knowledge

10.1 From retrieval to controlled interpretation

An AI assistant can locate evidence and summarize rules, but retrieved data is not authority for a policy change. A knowledge-grounded assistant should identify the current policy version, explain how facts map to conditions and flag missing or conflicting evidence. Action execution must remain behind explicit authorization and validation boundaries.

10.2 An order confirmation inquiry

User: “Can order 482 be confirmed?” The system retrieves state, customer eligibility and lines, evaluates the approved policy through a trusted decision path, then returns eligibility, reasons, uncertainties and references. The assistant does not invent line validity from the presence of rows and must not commit the order merely by phrasing a plausible explanation.

10.3 Measure groundedness, not readiness slogans

Measure the share of responses supported by cited policy versions, agreement with tested decisions, unresolved ambiguities and unsafe action attempts blocked. Do not mistake a label such as “AI-ready” for an operational success metric.

A Practical Modernization Pilot

11.1 Team and roles

A pragmatic pilot involves a business policy owner, a legacy-system specialist, a domain analyst, a solution architect, an engineer/test engineer and an operations or security representative. One person may cover several roles in a small organization, but approval authority and implementation responsibility must remain distinguishable.

11.2 Indicative sequence

A six-to-ten-week pilot is a planning hypothesis, not a delivery promise. Weeks 1–2: inventory and evidence collection; weeks 3–4: workshops, model and approved cases; weeks 5–7: decision-path prototype and differential testing; weeks 8–10: controlled rollout evaluation, review and go/no-go decision. Complex integrations or insufficient evidence may extend the schedule materially.

11.3 When not to use this approach

Avoid a knowledge-extraction-heavy initiative when the application is being retired without replacement, the business capability is trivial or standardized, no accountable domain expert can validate recovered semantics, or a fixed-price package replacement clearly dominates custom reconstruction. Where safety or regulatory risk is high, additional formal verification and assurance may be needed.

11.4 Pilot scorecard

Dimension	Evidence of progress
Coverage	Critical decisions discovered, classified and owned.
Traceability	Approved rules linked to evidence and acceptance cases.
Semantic quality	Differences resolved or consciously accepted.
Operability	Measured behavior, rollback and observability.
Transferability	Model reused in documentation, decisions and tests.

Table 11.1: Progress indicators for a modernization pilot.

Deliverable: Pilot dossier with measurable outcomes and explicit go/no-go decision.

Governance and Continuous Evolution

12.1 Keep knowledge independent of one codebase

Assign version identifiers, owners and review processes to business rules. Link each rule to affected applications and tests. A policy change should trigger impact analysis before deployment, and operational incidents should feed evidence back into the model.

12.2 Avoid a second undocumented legacy

A model becomes another legacy artifact if it drifts from execution. Automate checks wherever possible and periodically compare the model, current implementation and observed outcomes. Record exceptions rather than burying them in local patches.

12.3 The distinctive outcome

Modernization creates lasting value when the organization can discuss and change a business decision without first reverse-engineering its implementation. The reusable result is a governed understanding of business behavior, supported by evidence, tests and explicit operational contracts.

Business Knowledge Record Template

Field	Required contents
ID and title	Stable identifier; concise business assertion.
Scope	Capability, entities, channels and effective date.
Evidence	Source location, version, trace and interview references.
Evidence level	Candidate / Observed / Corroborated / Approved / Verified, scoped to the specific assertion.
Observation	What the implementation demonstrably does.
Intended policy	Business-approved statement and examples.
Ambiguities	Unresolved definitions and boundary cases.
Disposition	Required / tolerated / deprecated / defect.
Owner and lifecycle	Accountable approver; draft / approved / superseded.
Traceability	Model section, implementation contract and tests.

BR-ORD-014 Traceability Matrix

ID	Observation or policy	Explicit representation	Verification
BR-ORD-014	Draft, active customer, nonempty lines in legacy	CanConfirm with ValidLine as approved target	AC-014-01 allow valid line
BR-ORD-014-A	Legacy permits line count of one even if cancelled	Target excludes cancelled lines; approved behavior difference	AC-014-02 reject invalid-only order
BR-ORD-014-B	Customer activity checked	Eligibility requires active customer	AC-014-03 reject inactive customer
BR-ORD-014-C	Reservation failure and partial success unspecified	All-or-nothing reservation; release partial holds before confirmation	AC-014-04 partial compensation
BR-ORD-014-D	Mixture of valid and cancelled lines	Retain all order lines; reserve only valid lines (illustrative)	AC-014-05 mixed lines
BR-ORD-014-E	Repeat requests under retry conditions	Idempotency key returns prior decision without repeated side effects	AC-014-06 duplicate requests

Applying the Method with logiCells

C.1 The role of logiCells

logiCells promotes application modernization through the recovery, structuring and governance of business knowledge. This guide deliberately states the method without requiring a proprietary grammar or runtime. Unlike a generic documentation exercise, logiCells positions structured business knowledge and conceptual modeling as assets that can inform application behavior and its evolution; its published architectural direction includes conceptual models and a knowledge hypergraph.

The extent of tooling support and automation must be verified in a pilot. A logiCells engagement can use this framework to scope an application boundary, capture evidence, organize domain workshops, maintain rule traceability and plan incremental replacement. Those are proposed engagement activities, not assertions that any particular product automatically performs each step.

C.2 A practical engagement boundary

The most useful starting point is a small but consequential business decision, such as BR-ORD-014. The joint team identifies authoritative sources, makes ambiguities visible, secures business approval and establishes a testable contract before choosing implementation technology. The deliverable is a versioned, reviewable knowledge baseline with acceptance cases and an integration plan. Architectural and tooling choices remain explicit decisions, not prerequisites to understanding the method.

C.3 Contact and next step

For a scoped BR-ORD-014-style demonstration, visit <https://www.logicells.com/> or contact contact@logicells.com. Ask for a documented pilot that traces one business decision from legacy evidence through approval, model, test and controlled rollout.

C.4 What should be demonstrated before adoption

Ask for one end-to-end pilot showing a legacy trace, an approved decision table, a maintained knowledge model, acceptance and characterization tests, change-impact traceability and a measured rollout. Require evidence for tool-specific claims; this appendix intentionally makes no unverified claims about automated code analysis, code generation or execution.

Readiness Checklist

- ☐ Has each consequential business assertion an owner, evidence and version?
- ☐ Are legacy observations explicitly distinguished from target decisions?
- ☐ Have boundary scenarios been reviewed, including duplicates and failures?
- ☐ Are the knowledge model and operational contract understandable without reference to one vendor?
- ☐ Are tests linked to approved decisions, not merely to observed old behavior?
- ☐ Are technical limitations, uncertainty and rollout risks documented?
- ☐ Have diagrams, glossary and examples been reviewed by a business and technical representative?