IJC
International Journal of Computer (IJC)
Peer-Reviewed International Journal
ISSN: 2307-4523
https://ijcjournal.org

Dependency-Ordered Migration of Financial Transaction Data Between CRM and ERP Systems: A Reconciliation Protocol for Day-1 Ledger Integrity

Subhani Shaik

Enterprise Digital Transformation Data Architect, 130 Queso Place, Fremont, CA, 94539, USA

** Corresponding author: subhanishaik2452@gmail.com*

Accepted: 15 October 2026 | Published: 20 November 2026

Abstract

When enterprises consolidate customer and financial systems, whether through cloud transformation, merger, or divestiture, the migration of financial transaction data is the step most likely to compromise the first post-cutover financial close. Unlike master data, transactional records carry sequential and relational dependencies: a payment cannot exist without its invoice, an invoice without its customer and contract, and a trial balance is correct only if every open item lands in the right period, account, and currency with the right sign. Prevailing practice remains mapping-centric: it validates that columns were transformed correctly, not that the ledger still balances. In enterprise migration programs, conventional mapping-centric workflows are frequently accompanied by 10–15% late-stage or post-load manual remediation rates in financial data handling and risk delaying Day-1 revenue recognition. This paper presents Financial Object Synthesis (FOS), a system-agnostic protocol that models the financial data estate as a directed dependency graph, derives a strict load order from it (reference and entity master → contracts → open transactions → applications → historical ledgers), and gates every stage on formal parity conditions: trial-balance parity, subledger-to-general-ledger tie-out, and open-item parity. Across iterative mock cycles (typically 3–4 cycles plus pre-production rehearsal), load error/fallback rates are driven down monotonically, all loads execute idempotently so that quarantined records can be re-processed without duplication, and residual quarantined ledger exceptions are resolved prior to finance sign-off. We evaluate FOS through two retrospective, author-led enterprise case studies: a multi-billion-dollar technology divestiture migrating legacy enterprise resource planning (ERP) financials into a cloud ERP, and an information-services demerger separating a shared legacy estate. Across the mock-cycle cadence, practitioner-observed automated error/fallback rates declined from approximately 50% in Mock Cycle 1 to 30% in Mock Cycle 2, 10–15% in Mock Cycle 3, and below 0.5% in pre-production and production cutover, with residual quarantined exceptions resolved before sign-off to yield zero residual Day-1 trial-balance variance on in-scope accounts. Relative to non-contemporaneous historical client accounts and initial project plans, deployment was also associated with approximately 50% shorter reconciliation windows and build durations roughly 30% below initial plan estimates, although non-controlled baselines and author-led execution preclude strict causal attribution. We formalize the ordering rules and parity conditions, document the evaluation methodology and validity threats, and discuss limits including multi-currency revaluation, mid-period cutovers, and the need for independent external replication.

Keywords: financial data migration; ERP migration; trial balance reconciliation; Day-1 readiness; data integrity; dependency graph; cloud ERP

1. Introduction

Every enterprise system consolidation ends at the same place: the first financial close on the new platform. A merger closes, a divested business unit begins operating on its own systems, or a legacy enterprise resource planning (ERP) is retired in favor of a cloud platform, and within days the finance organization must produce a trial balance, an accounts-receivable aging, and a revenue figure that auditors will accept. If the migrated data does not support that close, the consequences are not technical. They are restatement risk, delayed reporting, covenant exposure, and in the case of a transaction, valuation decay. Research on post-merger integration consistently identifies information-system

integration as a leading cause of unrealized deal value [1], [2], and practitioner surveys have long reported that data migration projects overrun their schedules and budgets at high rates [3], [4].

Over eighteen years leading cloud data migration and master data management programs for Fortune 100 enterprises, I have observed that the failures in these programs are concentrated in one category of data: financial transactions. Customer master records, product catalogs, and even contract headers migrate with well-understood tooling. Invoices, payments, credit memos, journal lines, and the applications that link them do not. They are voluminous, interdependent, period-sensitive, and subject to accounting rules that the source and target systems interpret differently. Commercial migration tools treat them as rows to be mapped and loaded; finance treats them as a ledger that must balance.

This gap is structural. Mapping-centric tools, from vendor bulk loaders to general-purpose extract-transform-load (ETL) platforms, validate that each field was transformed according to a rule. They do not validate that the set of records, taken together, still satisfies the accounting identities the business depends on. The result is a class of defects that no field-level check can catch: a payment loaded before its invoice becomes an unapplied cash orphan; an open invoice loaded into a closed period silently distorts aging; a credit memo with a reversed sign inflates receivables; a rounding difference in currency conversion accumulates into a trial-balance variance that takes weeks to trace. In prior enterprise programs observed under conventional mapping-centric workflows (as well as late pre-stabilization mock runs prior to full gate closure), approximately 10–15% of financial records required manual remediation or fallback handling.

This paper presents Financial Object Synthesis (FOS), a protocol developed and applied across large-scale customer relationship management (CRM) and ERP migrations to systematically detect, quarantine, and resolve this class of defect prior to production cutover. FOS is the financial-data component of a broader migration framework, the Autonomous Semantic Harmonization Framework (ASHF), but it is self-contained and independent of any particular harmonization or mapping technique. Its contributions are:

- A formal model of the financial data estate as a directed dependency graph, and a definition of relational orphaning in terms of that graph.
- A load-ordering rule derived from the graph by topological sort, with explicit handling of many-to-many application objects that would otherwise create cycles.
- Three parity conditions, trial-balance parity, subledger-to-general-ledger tie-out, and open-item parity, expressed as stage gates with zero-tolerance policies for amounts and bounded tolerances for currency rounding.
- An iterative mock-cycle procedure (typically 3–4 mock cycles followed by pre-production rehearsal) with an explicit error/fallback-rate metric, convergence tracking, idempotent load semantics, and a quarantine mechanism that allows failed records to be re-processed without duplication.
- A retrospective empirical evaluation across two author-led enterprise programs that distinguishes directly observed cutover outcomes (automated fallback rates converging from approximately 50% in Mock Cycle 1 to below 0.5% in pre-production and production, with zero residual Day-1 trial-balance variance after quarantine resolution) from non-contemporaneous historical and plan-based comparisons, paired with a structured threats-to-validity analysis.

The remainder of the paper is organized as follows. Section 2 reviews related work in data migration, information-system integration in mergers and acquisitions, and data quality. Section 3 formulates the problem. Section 4 presents the protocol. Section 5 reports the evaluation. Section 6 discusses generalization and limitations, and Section 7 concludes.

2. Background and Related Work

2.1 Data migration as a discipline

Data migration has been treated as an engineering discipline in its own right for roughly two decades. Morris [4] established a practitioner methodology centered on business ownership of data, landscape analysis, and a formal migration strategy, and Haller [5] proposed reusable concepts and patterns for migrations into standard software packages, distinguishing between the migration of master and transactional data and noting that the latter is disproportionately error-prone. Matthes, Schulz, and Haller later formalized a detailed process model for large-scale migrations, with explicit phases for cleansing, mock migration, and cutover [6]. Bloor Research surveys of large enterprises found that migration projects consistently overran time and budget, with data quality and unforeseen complexity as leading causes [3]. Azeroual and Jha argue that without a data-quality program there is no data migration, and that quality must be measured and enforced during the migration rather than inspected afterward [7]. These works establish mock cycles and pre-load cleansing as good practice, but they do not formalize what a mock cycle must prove for financial data specifically, nor do they define the conditions under which a cutover may proceed.

2.2 Information-system integration in mergers and acquisitions

The management literature has examined why acquisitions fail to deliver expected value. Christofferson, McNish, and Sias attribute a large share of missed synergy targets to integration execution, including systems [1]. Tanriverdi and Uysal show empirically that cross-business IT integration capability is associated with acquirer value creation [8], and Henningsson, Yetton, and Wynne provide a comprehensive review of information-system integration in mergers and acquisitions, concluding that integration of transactional systems is the least understood and most consequential dimension [2]. None of this work addresses the mechanics of migrating financial transactions so that a combined or separated entity can close its books on Day 1; that is the gap this paper fills.

2.3 Data quality and reconciliation

Wang and Strong defined data quality in terms of dimensions meaningful to data consumers, including accuracy, completeness, consistency, and timeliness [9]. Batini and colleagues surveyed methodologies for assessing and improving data quality, noting that most methodologies assess attributes in isolation rather than relationships between records [10]; it is precisely the relationships between financial records that determine whether a ledger balances. Financial reporting requirements impose their own constraints: revenue recognition under Accounting Standards Codification (ASC) 606 and International Financial Reporting Standard (IFRS) 15 requires that contract, performance obligation, and transaction price data be traceable from the moment the target system is live [11]. The reconciliation conditions defined in this paper are designed to satisfy those requirements at cutover rather than to reconstruct them afterward.

2.4 Transactional foundations

The protocol draws on established engineering results. Helland's argument that idempotence is an essential property of reliable distributed operations underlies the load semantics in Section 4.6 [12]. Kimball and Caserta's ETL toolkit supplies the staging and audit patterns adapted for the stage gates [13]. Kahn's algorithm for topological sorting [14] is the basis for load ordering. Rahm and Bernstein's survey of schema matching [15] addresses the mapping problem, which is complementary to, and deliberately separated from, the ordering and reconciliation problem addressed here.

2.5 The remaining gap

Taken together, the literature provides process models for migration, evidence that IS integration determines deal value, a vocabulary for data quality, and the transactional primitives needed to build

reliable loaders. What it does not provide is a protocol that (a) defines the order in which financial objects must be loaded so that relational integrity is preserved by construction, (b) specifies the accounting identities that must hold at each stage, and (c) states when a migration is finished. FOS provides all three.

3. Problem Formulation

3.1 Classes of financial data

Let the financial data estate of a source system be partitioned into five classes. Master data M comprises customers, vendors, legal entities, and the chart of accounts. Reference data R comprises currencies, exchange rates, tax codes, payment terms, and fiscal calendars. Contract data C comprises sales contracts, subscriptions, and orders that give rise to revenue. Open transactions T_open comprise invoices, credit memos, payments, and journal entries that affect balances in periods not yet closed on the target. Historical transactions T_hist comprise closed-period transactions retained for audit, aging history, and analytics. Finally, applications A comprise the links that associate one transaction with another: a payment applied to one or more invoices, a credit memo applied against an invoice, or an intercompany elimination linking two entities. Balances B are not migrated as data; they are derived on the target from T_open and opening-balance journal entries, and the correctness of the migration is judged by whether the derived balances equal the source balances.

3.2 The dependency graph

Define a directed graph G = (V, E) whose vertices V are the financial object types across all classes and whose edges E encode the relation "must exist before". An edge (u, v) ∈ E means that an instance of v cannot be validly created on the target unless the instance(s) of u it references already exist. For example, (Customer, Invoice) ∈ E, (Invoice, Payment Application) ∈ E, and (Fiscal Period, Journal Entry) ∈ E. For a given migration scope, let the instance-level graph be G* = (V*, E*), where V* is the set of records and E* the set of foreign-key references between them. A record x ∈ V* is a relational orphan on the target if at least one of its referenced parents is absent at the time x is loaded:

$$O = \{\, x \in V^* : \exists\, (y, x) \in E^* \ \text{such that}\ y \notin L(t_x) \,\} \quad (1)$$

where L(t) denotes the set of records successfully loaded on the target at time t and t_x is the load time of x. The first requirement of the protocol is that O = ∅ at cutover. Mapping-centric approaches do not guarantee this because they order loads by table, not by dependency, and because load failures in a parent batch are not propagated to dependent batches.

3.3 Ledger integrity conditions

Relational integrity is necessary but not sufficient. Three accounting identities must also hold between source and target. Let S and D denote source and destination, let a index general-ledger accounts, p index fiscal periods, and c index currencies. Let Bal(X, a, p, c) denote the balance of account a at the end of period p in currency c on system X. Trial-balance parity requires:

$$\Delta TB(a, p, c) = Bal(D, a, p, c) - Bal(S, a, p, c) = 0 \quad \forall a, p, c \text{ in scope} \quad (2)$$

Two further identities complete the definition of ledger integrity. Subledger-to-general-ledger tie-out requires that, on the destination, the sum of open items in each subledger (receivables, payables) for a given period and currency equals the balance of the corresponding control account in the general ledger. Open-item parity requires that the population of open items, not merely their totals, matches between source and destination after key translation: the multiset of (customer, document, due date, amount, currency) tuples must be identical, so that aging buckets and customer statements are correct on Day 1. Together with (2), these conditions are stated at zero tolerance for amounts in transaction currency. A bounded tolerance applies only to functional-currency amounts that arise from conversion, where source and target may apply rounding at different points in the calculation; this is addressed in Section 4.4.

3.4 Cost of failure

The economic argument for enforcing these conditions before cutover rather than remediating afterward follows from the asymmetry of remediation cost. A defect found in a mock cycle is corrected by re-running a transformation or by a business owner amending a source record. The same defect found after go-live is corrected on a live ledger: a period may have to be re-opened, a controlled journal reversed, customer statements re-issued, and the auditor informed. In the engagements reported in Section 5, client finance controllers qualitatively estimated the unit effort of post-go-live remediation to exceed mock-cycle remediation by more than an order of magnitude (although formal activity-based costing was not instrumented), and practitioner experience across conventional mapping-centric workflows places late-stage or post-load manual remediation at roughly 10–15% of financial records. The protocol's objective is therefore to shift defect discovery and remediation upstream into iterative mock cycles—reducing pre-cutover load fallback rates below operational thresholds (<0.5%) and resolving quarantined ledger exceptions before finance sign-off—accepting additional mock-cycle effort in exchange for lower post-cutover ledger risk.

4. The Financial Object Synthesis Protocol

4.1 Overview

FOS is organized as a sequence of stage gates. Each stage loads one class of financial objects in dependency order, then evaluates a set of control conditions. A stage gate passes only if all conditions evaluate to zero variance (or within the bounded rounding tolerance for converted amounts). No subsequent stage begins until the current gate passes. The full sequence is executed across iterative mock cycles (typically three to four full mock cycles followed by a pre-production dress rehearsal) against refreshed source snapshots until the record-level error/fallback rate declines below an agreed pre-production threshold (<0.5%) and all residual quarantined exceptions affecting in-scope ledgers are resolved to zero trial-balance variance, at which point the sequence is executed against the production source under a change freeze. Table 1 summarizes the stages, their contents, and the conditions evaluated at each gate.

Table 1. Stage gates of the Financial Object Synthesis protocol

Stage	Objects loaded	Gate conditions	Typical failure prevented
G0 Reference	Fiscal calendar, currencies, exchange rates, tax codes, payment terms, chart of accounts	Referential completeness; period status parity; rate-table coverage of all transaction dates	Transactions posted to undefined periods or accounts; missing FX rate for a historical date
G1 Master	Legal entities, customers, vendors, bill-to/ship-to hierarchies	Record-count parity; key cross-reference completeness; duplicate-key check	Invoices with no customer; duplicate customers splitting receivables
G2 Contract	Contracts, subscriptions, orders, performance obligations, price books	Count and amount parity; contract-to-customer completeness; obligation-to-contract completeness	Revenue schedules without a contract; orphaned line items
G3 Open transactions	Open invoices, credit memos, unapplied payments, opening-balance journals	Trial-balance parity (Eq. 2); subledger tie-out; open-item parity; sign and period checks	Aging distortion; receivables overstated by sign error; unbalanced opening journal
G4 Applications	Payment-to-invoice applications, credit applications, intercompany links	Applied-amount parity per document; no application exceeding document amount; re-evaluation of G3 conditions	Unapplied cash; over-applied invoices; broken intercompany elimination
G5 History	Closed-period transactions and their applications, retained for audit and analytics	Count and amount parity per closed period; period-locked status on target	History posting into open periods; audit trail gaps
G6 Cutover	Delta extraction since freeze; final re-run of all gates	Pre-prod/prod fallback rate <0.5%; residual quarantined exceptions resolved to zero trial-balance variance; finance sign-off recorded	Transactions created during freeze lost; unsigned cutover

4.2 Load ordering by topological sort

The stage sequence in Table 1 is not an arbitrary convention; it is the topological order of the type-level dependency graph G. Given G, Kahn's algorithm [14] produces a linear ordering in which every vertex appears after all of its predecessors. Algorithm 1 states the procedure as applied to financial object types. Within a stage, the same procedure is applied at the instance level to G* so that, for example, a parent customer in a hierarchy is loaded before its child accounts, and a contract header before its lines.

Algorithm 1. Dependency-ordered load sequence
Input: type-level graph G = (V, E)
Output: ordered list Λ of object types
1. For each v ∈ V compute indegree(v) = |{ (u, v) ∈ E }|.
2. Initialize queue Q with all v such that indegree(v) = 0 (reference data).
3. While Q is non-empty:
a. Remove v from Q; append v to Λ.
b. For each (v, w) ∈ E: decrement indegree(w); if indegree(w) = 0, enqueue w.
4. If |Λ| < |V|, a cycle exists; apply the junction-object rule of Section 4.3 and repeat.
5. Group Λ into stages G0–G5 by class; load each stage in Λ order; evaluate the gate.

4.3 Many-to-many applications and cycle breaking

Financial data contains relationships that are many-to-many by nature: a single payment may settle several invoices, and a single invoice may be settled by several payments. Source systems represent this either with an explicit application (junction) object or with mutual references between payment and invoice. In the latter case G contains a cycle, and no topological order exists. The protocol resolves this by a junction-object rule: every many-to-many relationship is reified as an explicit application object whose only edges are inbound from the objects it links. Payments and invoices are then loaded independently in stage G3, with payments initially unapplied, and the application objects are loaded in stage G4. Assuming no other cyclic dependencies among parent classes, the rule ensures that G is acyclic, and it has a second benefit: the applied amount on each document becomes a derived quantity that can be checked at G4 against the source, rather than a stored attribute that may have been migrated inconsistently. The G4 gate therefore requires, for every open document, that the applications loaded against it sum exactly to the applied amount recorded on the source and that this sum does not exceed the document amount; any document failing either check is quarantined together with its applications.

4.4 Control totals and tolerance policy

Each gate evaluates a variance vector rather than a single total. For stage G3, the vector Δ is indexed by account, period, and currency as in (2); for G5 by closed period; for G1 and G2 by entity and object type. Evaluating variance at this granularity is what makes discrepancy reports actionable: a non-zero trial-balance variance at (a, p, c) points directly to the account, period, and currency in which records are missing or mis-signed, and the offending records can be enumerated by set difference between source and target at that index. Amounts in transaction currency are held to zero tolerance. For functional-currency amounts that the source and destination compute by conversion, a bounded tolerance is permitted to absorb differences in rounding order. With N(a, p, c) transactions contributing to a balance and a minor unit u for the functional currency, the tolerance is:

$$| \Delta func(a, p, c) | \le \tau(a, p, c) = N(a, p, c) \cdot u / 2 \qquad (3)$$

This bound is the maximum accumulated half-unit rounding difference and is typically a few currency units on balances of millions. Any variance exceeding τ is treated as a defect. Tolerance is never applied to record counts, document amounts, or open-item populations.

4.5 Mock cycles and convergence

A mock cycle is a complete execution of stages G0–G5 against a point-in-time snapshot of the source, followed by evaluation of every gate and generation of a discrepancy report. For each cycle k, let N_err,k denote the count of records rejected by schema, referential, or parity checks or routed to manual

quarantine/fallback handling, and let N_tot,k denote the total population of in-scope financial records processed in cycle k. The cycle error/fallback rate is defined as Ek = N_err,k / N_tot,k (Eq. 4). Between cycles, discrepancies are resolved by one of three actions: correction of source data by the business owner, correction of transformation or key-translation logic by the migration team, or an explicit, documented accounting disposition (for example, how to treat a legacy write-off category that has no target equivalent). Across a typical enterprise cadence of three to four mock cycles followed by a pre-production dress rehearsal, execution continues until the cutover readiness condition holds:

$$E_k = N_{err,k} / N_{tot,k} \quad (4)$$

$$E_{preprod} < 0.5\% \quad and \quad \Delta TB_{post\text{-}quarantine}(a, p, c) = 0 \quad \forall a, p, c \in scope \quad (5)$$

Condition (5) separates automated load convergence from final ledger sign-off. In multi-million-record enterprise estates, a small tail of residual source-side exceptions (<0.5% fallback rate) routinely persists into pre-production and production runs; rather than assuming raw first-pass errors vanish completely, FOS requires that the automated fallback rate fall below 0.5% and that every remaining quarantined record affecting in-scope balances be resolved (via idempotent re-submission or controller-approved disposition) so that post-remediation trial-balance variance reaches zero prior to sign-off. Progress across cycles is tracked by the cycle-over-cycle convergence ratio (Ek / Ek−1). Persistent stagnation after the second cycle indicates record-by-record patching rather than root-cause rule or source remediation and triggers an engineering review. In practice, Mock Cycle 1 surfaces the broadest set of structural and mapping defect classes, Mock Cycles 2 and 3 resolve rule-level causes and G4 application logic, and an optional fourth mock cycle or pre-production dress rehearsal clears residual business-owned exceptions. For illustration, on a hypothetical 1,000,000-record financial estate, an automated fallback progression of ~50% → ~30% → 10–15% → <0.5% reduces the exception pool from 500,000 records in Mock Cycle 1 to fewer than 5,000 quarantined records in pre-production—a tractable tail for targeted pre-sign-off resolution compared with 100,000–150,000 post-load exceptions under a 10–15% conventional defect rate.

4.6 Idempotent loads and quarantine

Every load in the protocol is an idempotent upsert keyed on a stable external identifier that is carried from source to target and persisted on the target record [12]. This has three consequences. First, a mock cycle can be re-run against the same target without truncation, and only changed records are affected. Second, a record that fails a gate is not discarded but placed in a quarantine set Q with the condition it failed; after correction it is re-submitted and, because the load is idempotent, cannot create a duplicate. Third, the delta load at cutover (Section 4.7) is simply another idempotent pass over records changed since the freeze timestamp. All quarantined records affecting in-scope ledger balances must be resolved—via source correction, rule remediation, idempotent re-submission, or documented controller disposition—so that residual trial-balance variance reaches zero as required by condition (5).

4.7 Cutover and sign-off

When condition (5) is satisfied in pre-production rehearsal, the production cutover proceeds under a change freeze on the source. A final full execution of G0–G5 is performed against the frozen source, followed by a delta extraction of any records modified after the last rehearsal snapshot. All gates are re-evaluated, and any residual fallback records (<0.5%) are resolved through quarantine or documented disposition before cutover closes. The finance controller for each legal entity reviews the gate results and records a sign-off against the trial-balance parity report before the target is opened for transactions. The sign-off artifacts (variance reports at zero after quarantine resolution, open-item parity report, quarantine disposition log, and applied-amount report) serve as opening-balance evidence for the first audit on the new platform. Because the source remains operational throughout mock cycles and the final

freeze is bounded to hours rather than multi-day shutdown windows, the protocol is designed to avoid extended operational downtime during cutover.

5. Evaluation

5.1 Case settings, data scope, and confidentiality boundaries

We evaluate FOS through a retrospective, observational multi-case industrial study (N = 2) drawn from two enterprise transformation programs led by the author through a global consulting firm between 2017 and 2018. Because both programs were live commercial engagements rather than controlled experiments, and both were delivered under the author's technical leadership, this evaluation is structured to report directly observed cutover outcomes separately from non-contemporaneous comparative baselines and to bound causal claims accordingly. No independent empirical case conducted by teams unaffiliated with the protocol developer is included in this study.

Case A involved the divestiture and carve-out of a multi-billion-dollar software business from a global technology enterprise, requiring separation of financial records from a shared legacy ERP and migration into a cloud ERP and cloud CRM for the newly independent entity, which was required to report Day-1 revenue and balance sheet independently. Execution ran from March through August 2017 (approximately six months) across an approximate multi-million-record total data volume (on the order of 10^6–10^7 financial records across stages G0–G5) spanning multiple legal entities and currencies. Case B involved the demerger of an information-services business from a larger media and information group between October 2017 and June 2018 (approximately eight to nine months) as part of a broader cloud transformation, migrating a multi-entity financial estate on the order of 10^6 total records across stages G0–G5 from legacy ERP and CRM platforms into cloud ERP and CRM while rationalizing and retiring redundant shared legacy datasets.

In both engagements, the migrated financial population encompassed all six protocol classes (G0–G5): reference master tables (fiscal calendars, charts of accounts, currencies, exchange-rate tables, tax codes, payment terms), party and organizational master hierarchies (legal entities, customers, vendors, bill-to/ship-to structures), commercial contracts and order schedules, open subledger and general-ledger transactions (open AR/AP invoices, credit memos, unapplied cash/payments, and opening-balance journals), many-to-many payment and credit application junctions, and closed-period historical transactions retained for audit and aging traceability. Under client non-disclosure agreements (NDAs) and retrospective engagement archival policies from 2017–2018, exact proprietary record counts per object table, raw monetary balances, and exported per-cycle client log files cannot be disclosed or independently audited. Table 2 summarizes the engagement settings, migrated object scope, mock-cycle cadence, and data-availability boundaries.

Table 2. Case settings, migrated financial object scope, and archival availability

Attribute	Case A: Technology divestiture	Case B: Information-services demerger
Study design & role	Retrospective industrial case; author-led delivery	Retrospective industrial case; author-led delivery
Transaction type	Carve-out and merger of divested software unit	Demerger and cloud ERP/CRM transformation
Source platforms	Shared legacy ERP (financials, order management)	Legacy ERP and legacy CRM
Target platforms	Cloud ERP (financials) and cloud CRM	Cloud ERP and cloud CRM
Actual execution period	March–August 2017 (~6 months)	October 2017–June 2018 (~8–9 months)
Migrated object scope (G0–G5)	Approx. 10^6–10^7 total financial records (order-of-magnitude volume across G0–G5; exact per-table counts NDA-restricted): reference/COA, multi-entity customers & vendors, contracts/orders, open AR/AP invoices, credit memos,	Approx. 10^6 total financial records (order-of-magnitude volume across G0–G5; exact per-table counts NDA-restricted): multi-entity reference/COA, master, contracts, open/historical transactions & applications,

Attribute	Case A: Technology divestiture	Case B: Information-services demerger
	payments, opening journals, payment/credit applications, and closed-period history	plus profiling and retirement of redundant shared legacy datasets
Mock & rehearsal cadence	3–4 iterative mock cycles + pre-production rehearsal + production cutover (G6)	3–4 iterative mock cycles + pre-production rehearsal + production cutover (G6)
Day-1 reporting obligation	Independent revenue recognition and opening balance sheet for divested entity	Independent statutory and management financial reporting for demerged entity
Raw-data disclosure constraint	Exact record counts, monetary totals, and raw 2017 cycle logs restricted under client NDA / archival retention	Exact record counts, monetary totals, and raw 2017–2018 cycle logs restricted under client NDA / archival retention

5.2 Evaluation design, metric definitions, and baseline data sources

Five operational metrics were tracked across the engagements. Because equivalent contemporaneous control groups were not feasible in live corporate divestitures and demergers, each metric below specifies its operational definition, data source for the comparative baseline, and whether the comparison supports observational association only rather than causal attribution:

Day-1 residual trial-balance variance (directly observed cutover outcome): Count of non-zero (account, period, currency) cells in ΔTB at Stage G6 sign-off after applying the functional-currency rounding bound in Eq. (3) and resolving quarantined fallback exceptions. Comparative context comes from prior conventional migrations at the client organizations, where post-close adjusting journals were routinely required; no contemporaneous control group was run.

Migration error / fallback rate, Ek (practitioner-observed programmatic progression): Defined formally in Eq. (4) as (errored or fallback records) / (total in-scope records) for cycle k, capturing records rejected by target validations, failing referential or parity gates, or routed to quarantine/fallback handling. The 10–15% conventional benchmark reflects practitioner-observed late-stage or post-load manual remediation rates in prior mapping-centric migrations within the same organizations and aligns with the error rate still observed at Mock Cycle 3 prior to pre-production gate stabilization. Because raw client remediation logs from historical pre-FOS migrations were not archived under NDA, this 10–15% range is reported as an approximate programmatic baseline rather than an audited control measurement.

Financial reconciliation time (historical client-reported comparison): Defined as elapsed calendar duration from completion of the final financial load batch to legal-entity finance controller sign-off on opening trial-balance and subledger tie-out reports. The comparative baseline was drawn from retrospective client-finance accounts of prior migrations on the same legacy estates. Because historical client reconciliations were not instrumented with automated timestamp logging comparable to FOS gate execution, the observed ~50% shorter window is an approximate historical comparison rather than a controlled timing experiment.

Mock-cycle throughput (Case B within-team observation): Relative volume of financial data processed per mock cycle under constant delivery team headcount in Case B, comparing later protocol-governed mock cycles against initial pre-protocol cycle throughput by the same team. This metric was not formally tracked in Case A (reported as n/m).

Migration build timeline versus plan (plan-based baseline comparison): Elapsed calendar duration from migration design initiation to production cutover (Case A: ~6 months actual; Case B: ~8–9 months actual), compared against each program's initial Statement-of-Work (SOW) and Work-Breakdown-Structure (WBS) baseline schedule prepared using conventional migration estimating assumptions. Because SOW/WBS plans are forward estimates that incorporate risk contingencies, staffing assumptions, and initial scope expectations, finishing ~30% ahead of plan cannot be causally attributed solely to FOS.

5.3 Observed outcomes

Across both retrospective engagements, executing FOS over 3–4 iterative mock cycles and a pre-production dress rehearsal was associated with progressive reduction of load fallback rates and zero residual Day-1 trial-balance variance on in-scope accounts at controller sign-off. Specifically, automated error/fallback rates declined from approximately 50% in Mock Cycle 1 to approximately 30% in Mock Cycle 2, 10–15% in Mock Cycle 3, and below 0.5% in pre-production rehearsal and production cutover. Rather than assuming zero first-pass record errors across millions of records, the residual <0.5% fallback population was isolated in quarantine and cleared via source remediation, rule fixes, idempotent re-submission, or documented controller disposition prior to cutover sign-off, so that finance teams in both programs signed off on opening trial balances without requiring post-close adjusting journals.

Compared against retrospective client accounts of prior migrations, load-to-sign-off reconciliation time was approximately 50% shorter in both engagements—an outcome stakeholders associated primarily with granular (account, period, currency) variance vectors that enabled review by exception rather than manual sampling. In Case B, the same delivery team processed approximately 2.5× the financial data volume per cycle after protocol adoption, and G0–G1 metadata profiling surfaced redundant shared legacy datasets that were decommissioned prior to migration rather than carried into the cloud target. Both engagements completed build-to-cutover execution approximately 30% below their initial SOW/WBS planned timelines and avoided multi-day source-system shutdowns by confining production cutover to a bounded change-freeze window measured in hours. Table 3 summarizes all observed outcomes alongside their baseline data sources and explicit attribution caveats.

Table 3. Observed outcomes versus comparative baselines and attribution caveats

Metric	Comparative baseline & source	Case A (observed)	Case B (observed)	Attribution status & caveat
Day-1 residual trial-balance variance (non-zero ΔTB cells)	Qualitative client history: typically >0 cells requiring post-close adjustments	0 non-zero cells at G6 sign-off (post-quarantine)	0 non-zero cells at G6 sign-off (post-quarantine)	Directly observed at cutover; no contemporaneous control group
Automated error / fallback rate, Ek (Eq. 4)	Approx. 10–15% post-load manual remediation in prior migrations (also Mock 3)	~50% (M1) → ~30% (M2) → ~10–15% (M3) → <0.5% (pre-prod/prod); resolved pre-sign-off	~50% (M1) → ~30% (M2) → ~10–15% (M3) → <0.5% (pre-prod/prod); resolved pre-sign-off	Practitioner-observed programmatic rates; raw 2017–2018 logs NDA-restricted
Financial reconciliation time (load to sign-off)	Retrospective client-finance reports of prior migrations on same estate	Approx. 50% shorter than client-reported baseline	Approx. 50% shorter than client-reported baseline	Associational only; prior client baselines were uninstrumented estimates
Data volume per mock cycle (constant team)	Initial pre-protocol mock cycle cadence by same team (1.0×)	n/m	Approx. 2.5× relative throughput	Within-team observation in Case B only; partially reflects team learning
Build-to-cutover timeline vs. initial plan	Initial SOW/WBS baseline plan under conventional estimating assumptions	Approx. 30% below plan (~6 mo: Mar–Aug 2017)	Approx. 30% below plan (~8–9 mo: Oct 2017–Jun 2018)	Plan-vs-actual variance; initial plans are estimates with contingencies (non-causal)
Redundant legacy datasets identified & retired	Not formally assessed in conventional mapping scope	n/m	Identified at G0–G1 and retired prior to load	Directly observed operational outcome in Case B
Cutover disruption to source systems	Conventional cutover windows (hours to multiple days of downtime)	Bounded change freeze (hours); no multi-day shutdown	Bounded change freeze (hours); no multi-day shutdown	Directly observed operational procedure; non-controlled comparison

n/m: not formally measured in that engagement.

5.4 Mock-cycle convergence and discrepancy taxonomy

Table 4 details the practitioner-observed fallback-rate progression across the typical 3–4 mock-cycle cadence plus pre-production dress rehearsal and production cutover, together with the primary discrepancy classes surfaced at each stage. In Mock Cycle 1, enforcing strict dependency ordering and ledger gates immediately exposed broad structural defects—yielding an error/fallback rate of approximately 50% as unmapped reference codes, missing historical foreign-exchange (FX) rates, duplicate master keys, sign/polarity reversals, and orphaned child records were rejected rather than

silently loaded. Rule-level transformation and mapping corrections reduced the fallback rate to approximately 30% in Mock Cycle 2 (cycle-over-cycle ratio ~0.60) and to 10–15% in Mock Cycle 3 (ratio ~0.33–0.50). Notably, the 10–15% residual rate at Mock Cycle 3 mirrors the remediation burden frequently carried into production under schedule-driven conventional migrations; under FOS, an additional stabilization cycle (Mock Cycle 4 where needed) and pre-production rehearsal drove automated fallback below 0.5%, leaving only a small quarantined tail of edge-case source exceptions that were cleared or formally dispositioned before finance sign-off at zero residual trial-balance variance.

Table 4. Programmatic mock-cycle fallback progression and detected discrepancy taxonomy

Cycle / phase	Error / fallback rate (Ek)*	Cycle ratio	Dominant discrepancy classes detected (G0–G5) & primary resolution activity
Mock Cycle 1	~50%	—	G0 missing FX rates & unmapped fiscal periods/tax codes; G1 duplicate/unlinked customer & vendor keys; G3 sign/polarity reversals, unbalanced opening journals, parent-absent orphans. Resolution: core schema, key-translation, COA, and sign-convention fixes.
Mock Cycle 2	~30%	~0.60	G2 contract-to-customer & performance-obligation linkage gaps; G3 open-item period/aging mismatches & functional-currency FX rounding exceeding τ (Eq. 3); residual master hierarchy errors. Resolution: rule-level transformation fixes, period mapping, FX table completion.
Mock Cycle 3 (and optional Cycle 4)	~10–15%	~0.33–0.50	G4 many-to-many payment/credit application over- or under-allocations & unapplied cash mismatches; G5 closed-period history misclassifications; business-owned source anomalies. Resolution: G4 junction logic, business source cleansing, documented accounting dispositions.
Pre-Production (Rehearsal)	<0.5%	<0.05	Isolated residual source-data edge cases and late-arriving master/reference updates caught by G1–G4 quarantine. Resolution: targeted quarantine remediation, idempotent re-submission, and pre-cutover verification of ΔTB = 0 (Eq. 5).
Production Cutover (G6)	<0.5% → ΔTB = 0	Stable (<0.5%)	Minor freeze-window delta anomalies and isolated edge-case fallbacks caught in quarantine prior to release. Resolution: delta load, idempotent reprocessing or controller-approved disposition, and formal Day-1 finance sign-off.

**Note: Error/fallback rates follow Eq. (4), Ek = (errored or fallback records) / (total in-scope records), and represent approximate practitioner-observed programmatic figures across the 3–4 mock-cycle cadence rather than raw per-case exported logs, which remain restricted under client NDAs.*

5.5 Comparison with mapping-centric practice

Table 5 contrasts the protocol with the mapping-centric approach embodied in vendor bulk loaders and general-purpose ETL tooling. The distinction is not one of tooling quality; the same loaders and ETL platforms can be, and were, used to execute the loads within FOS. The distinction lies in what is verified, in what dependency order, and under what cutover readiness criteria.

Table 5. Mapping-centric practice versus the FOS protocol

Dimension	Mapping-centric practice	FOS protocol
Unit of verification	Individual field transformation	Accounting identity across the complete record set
Load order	By table, often alphabetical or by size	Topological order of the dependency graph
Many-to-many links	Migrated as stored attributes	Reified as junction objects loaded in G4; applied amounts derived and checked
Trial balance	Checked after go-live by finance	Stage-gate condition at G3, G4, and G6 (zero transaction-currency tolerance; bounded FX rounding τ)
Failed / fallback records	Logged; often discarded or manually re-keyed post-go-live	Quarantined with failing condition; resolved and re-submitted idempotently pre-sign-off
Cutover readiness rule	Schedule-driven stopping point	3–4 mock cycles + pre-prod fallback <0.5% with quarantined ledger exceptions resolved to ΔTB = 0 (Eq. 5)
Evidence for audit	Reconstructed after cutover	Signed pre-cutover variance, open-item parity, applied-amount, and quarantine-disposition reports
Source cutover window	Extended cutover window (hours to days of downtime)	Bounded change freeze measured in hours

5.6 Threats to validity and methodological limitations

Because this evaluation relies on two retrospective, author-led enterprise engagements rather than controlled experiments, four threats to validity constrain how the empirical evidence should be interpreted:

Internal validity and causal attribution: Neither engagement included a contemporaneous control migration running conventional mapping-centric processes on the same estate. Comparative baselines for reconciliation time (prior client migrations) and build timeline (~30% below initial SOW/WBS estimates) are historical or plan-based and therefore susceptible to confounding from executive sponsorship, upfront finance participation, contingency buffers built into initial consulting plans, scope refinements, and team maturation across mock cycles. Accordingly, observed reductions in reconciliation duration and build schedule represent observational associations rather than isolated causal proof of FOS effectiveness.

Author/developer bias and absence of an independent case: Both cases were delivered by teams led by the protocol developer, creating risk that outcomes partially reflect domain expertise, delivery leadership, or Hawthorne effects rather than the protocol alone. While progressive within-program fallback reduction across mock cycles (50% → 30% → 10–15% → <0.5%) indicates operational mechanics within the same delivery teams, this paper does not present an independent empirical case study executed and instrumented by teams unaffiliated with the author. Subsequent adoption of protocol elements within the delivering firm's methodology (Section 6.4) is noted strictly as contextual uptake, not independent empirical validation.

Construct validity, measurement asymmetry, and archival constraints: FOS stage gates produced structured exception and parity tracking during execution, whereas client-reported prior reconciliation durations and historical 10–15% remediation benchmarks were derived from retrospective, uninstrumented client accounts. Moreover, client NDAs and retrospective archival constraints from 2017–2018 prohibit publishing exact per-table record populations, raw monetary balances, or raw cycle log files. All comparative percentages should therefore be read as approximate practitioner-reported programmatic figures rather than audited micro-level telemetry.

External validity and domain scope: With N = 2 engagements—both in technology and information-services enterprises characterized primarily by conventional receivables-driven accounting—the empirical scope does not establish generalizability to domains with structurally distinct ledger models (such as high-volume retail banking, insurance policy-reserve accounting, or complex manufacturing inventory valuation). Technical boundary conditions such as multi-currency revaluation timing, mid-period cutovers, and multi-wave intercompany consolidations are examined further in Section 6.5.

6. Discussion

6.1 Generalization across platforms

Nothing in the protocol depends on a specific source or target. The dependency graph is constructed from the target platform's object model and the source platform's foreign-key structure; the parity conditions are accounting identities that hold in any double-entry system. The protocol has been executed with legacy on-premises ERP and CRM sources and cloud ERP and CRM targets from several major vendors, using each vendor's native bulk-load interface for the physical loads. The junction-object rule of Section 4.3 is the only element that occasionally requires platform-specific work, because some target platforms expose payment applications as a first-class object and others expose them only through an API that applies cash as a side effect; in the latter case the application object exists in the staging layer only, and the G4 gate is evaluated by reading back the applied amounts from the target.

6.2 Organizational implications

The protocol changes who owns the migration. Because each gate requires finance sign-off on a variance report, the finance controller becomes a participant in every mock cycle rather than a recipient of the outcome. In both programs this was initially resisted as additional workload and subsequently valued, because the same reports served as opening-balance evidence for the first audit. The requirement that business owners correct source data, rather than the migration team patching it in transformation logic, likewise shifts effort earlier and makes it visible. Program leaders should plan for a longer preparation and mock phase and a shorter post-cutover stabilization phase; in both observed engagements, total build-to-cutover elapsed duration remained below initial project plans, though front-loaded finance participation was essential.

6.3 Cost

The protocol adds cost in three places: construction and maintenance of the dependency graph; execution of additional mock cycles; and the staging infrastructure required for idempotent loads and quarantine. In the programs reported, the additional mock cycles were the largest component. Against this, neither observed engagement required the extended post-cutover ledger stabilization period or multi-day cutover shutdowns typically budgeted under conventional approaches. Based on stakeholder assessments and the qualitative disparity between mock-cycle and post-go-live remediation effort (Section 3.4), program leaders judged this trade-off favorable in both cases, although formal activity-based cost accounting was not performed and remains a topic for instrumented future study.

6.4 Institutionalization

Following the two engagements reported here, elements of the protocol were incorporated into the delivering firm's migration methodology and referenced in enterprise transformation proposals. While this practitioner uptake suggests perceived operational utility, independent empirical telemetry from subsequent engagements led by other teams has not been formally collected or analyzed for this paper and therefore cannot be cited as independent empirical validation. The financial-data protocol is the component of that framework with the most direct bearing on the outcome executives care about, which is whether the books close.

6.5 Limitations

Three limitations should be stated. First, multi-currency estates in which the source and target apply revaluation at different points (transaction date versus period end) can produce functional-currency differences that exceed the rounding tolerance of (3) for legitimate reasons; the protocol handles this by requiring an explicit revaluation journal on the target and evaluating parity after it is posted, but this requires accounting judgment that the protocol cannot automate. Second, mid-period cutovers, in which the target must take over part-way through a fiscal period, require the open-transaction stage to be split by transaction date and the opening-balance journal to be constructed at an intra-period point; the parity conditions still apply but the source balances must be computed as of the cutover date rather than a period end. Third, intercompany eliminations across entities that are migrated in different waves cannot be fully verified until all entities are on the target; the protocol evaluates the intercompany subset of (2) at each wave and defers the consolidated check to the final wave.

7. Conclusion and Future Work

Financial transaction data is the component of an enterprise migration most likely to compromise Day-1 financial close and least well served by prevailing tooling, because that tooling validates fields rather than ledgers. This paper has presented FOS, a protocol that models the financial estate as a dependency graph, loads it in topological order with many-to-many applications reified and loaded after open transactions, gates each stage on trial-balance parity, subledger tie-out, and open-item parity,

iterates mock cycles to explicit fallback and parity readiness criteria, and uses idempotent loads with quarantine so that corrections cannot introduce duplicates. Across two retrospective, author-led enterprise case studies—a technology divestiture and an information-services demerger—applying FOS across 3–4 mock cycles and pre-production rehearsal was associated with automated fallback-rate convergence from approximately 50% in Mock Cycle 1 to 30% in Mock Cycle 2, 10–15% in Mock Cycle 3, and below 0.5% in pre-production and production cutover, with residual quarantined exceptions resolved prior to finance sign-off to yield zero residual Day-1 trial-balance variance on in-scope accounts. Relative to non-contemporaneous historical client accounts and initial SOW/WBS project plans, these engagements also exhibited approximately 50% shorter load-to-sign-off reconciliation windows and build timelines roughly 30% below initial plan estimates, without extended multi-day source-system shutdowns; however, non-controlled baselines, confidentiality constraints on raw archival logs, and author-led execution preclude strict causal attribution.

Four directions for future work follow. First, independent multi-site empirical evaluation—conducted by delivery teams unaffiliated with the protocol developer using standardized automated telemetry, contemporaneous baseline instrumentation, and shareable or synthetic benchmark datasets—is essential to test causal effectiveness and external validity beyond the two author-led cases reported here. Second, extending the dependency graph and parity conditions to subscription revenue under ASC 606 and IFRS 15 is needed where performance obligations and revenue schedules introduce time-phased objects whose parity must be evaluated per recognition period rather than per posting period. Third, formalizing the multi-entity consolidation case would allow intercompany parity to be evaluated incrementally across migration waves with clearer guarantees for the final consolidated ledger. Fourth, publishing a reference implementation of the gate evaluations as platform-independent SQL over a standard staging schema would support repeatable benchmarking and tool-independent adoption.

References

[1] S. Christofferson, R. McNish, and D. Sias, "Where mergers go wrong," *McKinsey Quarterly*, no. 2, pp. 92–99, 2004.

[2] S. Henningsson, P. W. Yetton, and P. J. Wynne, "A review of information system integration in mergers and acquisitions," *Journal of Information Technology*, vol. 33, no. 4, pp. 255–303, 2018. doi:10.1057/s41265-017-0051-9

[3] P. Howard, *Data Migration*. Bloor Research, 2011.

[4] J. Morris, Practical *Data Migration*, 2nd ed. Swindon, UK: BCS, The Chartered Institute for IT, 2012.

[5] K. Haller, "Towards the industrialization of data migration: Concepts and patterns for standard software implementation projects," in *Advanced Information Systems Engineering (CAiSE 2009), Lecture Notes in Computer Science*, vol. 5565. Berlin: Springer, 2009, pp. 63–78. doi:10.1007/978-3-642-02144-2_10

[6] F. Matthes, C. Schulz, and K. Haller, "A detailed process model for large scale data migration projects," in *Business Information Systems (BIS 2012), Lecture Notes in Business Information Processing*, vol. 117. Berlin: Springer, 2012, pp. 165–176. doi:10.1007/978-3-642-30359-3_15

[7] O. Azeroual and M. Jha, "Without data quality, there is no data migration," *Big Data and Cognitive Computing*, vol. 5, no. 2, art. 24, 2021. doi:10.3390/bdcc5020024

[8] H. Tanriverdi and V. B. Uysal, "Cross-business information technology integration and acquirer value creation in corporate mergers and acquisitions," *Information Systems Research*, vol. 22, no. 4, pp. 703–720, 2011. doi:10.1287/isre.1090.0250

[9] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, 1996.

[10] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Computing Surveys*, vol. 41, no. 3, art. 16, 2009. doi:10.1145/1541880.1541883

[11] Financial Accounting Standards Board, *Accounting Standards Update No. 2014-09, Revenue from Contracts with Customers (Topic 606)*. Norwalk, CT: FASB, 2014.

[12] P. Helland, "Idempotence is not a medical condition," *ACM Queue*, vol. 10, no. 4, pp. 30–46, 2012. doi:10.1145/2181796.2187821

[13] R. Kimball and J. Caserta, *The Data Warehouse ETL Toolkit*. Indianapolis, IN: Wiley, 2004.

[14] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM*, vol. 5, no. 11, pp. 558–562, 1962.

[15] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001.