SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Data Reconciliation Methodology for Enterprise Data Migration and Conversion Accuracy

Sridhar Gajavelli

Independent Researcher, TX,USA

ABSTRACT
Life insurers periodically retire legacy policy administration systems in favor of modern platforms, moving an entire in-force block of business, its coverages, and every party attached to each policy from one data model to another. A single migration can involve tens of thousands of policies and hundreds of thousands of individually mapped attribute values, far more than manual spot-checking can verify. This paper describes a data validation reconciliation methodology by using Business Intelligence tools. The methodology compares source and target extracts record by record and attribute by attribute after applying canonical transformation rules, classifies every comparison as a match, a mismatch, or a missing value on either side, and aggregates the results into per-attribute and per-record validation scores refreshed daily.

KEYWORDS: Business intelligence, data migration, data quality, data validation, ETL testing, insurance policy administration systems, Power BI, reconciliation, star schema.

I. INTRODUCTION

Insurance carriers that still administer part of their in-force business on older policy administration systems face a recurring decision: keep maintaining a system that is difficult to extend or migrate the block of business to a newer platform [1, 2]. A core system migration of this kind is not a simple file copy. Every policy record, every coverage or rider attached to it, and every party in a role on the policy, meaning the insured, the owner, the payer, the agent, and each beneficiary, has to be re-expressed in the target system's data model, often under a different set of codes, field names, and business rules than the source used.

This paper describes the validation approach used on one such migration, in which a block of whole life business was moved from a legacy administration system, referred to here as the source system, to a modern policy administration platform, referred to here as the target system. The company and specific product names involved are not the point of this paper and are withheld; what is described is the reconciliation methodology built to confirm, attribute by attribute, that what came out of the target system after conversion matches what went in from the source system, after accounting for the transformation rules that were supposed to be applied along the way.

The methodology draws on a principle laid out by Sureddy and Yallamula in their paper on debugging data warehousing and business intelligence platforms: that raw signals, whether log lines or, in this case, raw attribute values, are not useful on their own until they are correlated back to a known structure [3]. In their paper, that structure is the application and server architecture behind a live data warehouse, and the raw signals are error messages [4]. In a core system migration, the structure is the set of transformation and mapping rules that define how a source value is supposed to become a target value, and the raw signals are the actual values sitting in the source and target extracts [5]. Section II describes that connection in more detail. Sections III and IV describe the validation problem and the tool built to address it. Section V works through an actual defect the tool caught, involving how beneficiary designations are represented in the target system. Section VI reviews the validation results captured to date, Section VII discusses the testing effort the tool saves, Section VIII discusses the approach's limitations, and Section IX concludes.

II. RELATED METHODOLOGY

Sureddy and Yallamula's debugging methodology is built around the idea that an operations team facing an unfamiliar issue in a data warehouse or business intelligence platform needs a working knowledge of the server, application server, application tool, and data layers, and needs to know how to trace a symptom (an

error message, a failed batch job, a business user complaint) back through those layers to a root cause. Their methodology is reactive: it starts from a reported problem and works backward through logs and architecture. Core system migration validation is a related but different problem. Rather than waiting for a policyholder or a business user to report that something looks wrong, the validation tool described in this paper is meant to find migration defects before they are ever seen downstream, by comparing the full population of migrated records against the source system on a fixed schedule. The starting point is not a symptom but a transformation rule: for every attribute that is supposed to move from the source system to the target system, is the value that actually landed in the target system consistent with what the rule says it should be, for every record, not just a sample? The debugging methodology's core insight, that a raw value only becomes meaningful once it is checked against a known structure, still applies [6]. Here, the known structure is a set of canonical mapping tables (described in Section IV-C), and the raw values are the source and target extracts themselves.

This taxonomy also sits within the broader data quality literature. The four-way Match, Mismatch, Missing Source Data, and Missing Target Data classification introduced in Section IV-D is, in Wang and Strong's terms, a working operationalization of two specific data quality dimensions: accuracy, whether the target value agrees with the source once both are expressed canonically, and completeness, whether a value exists on both sides at all [7]. Sureddy and Yallamula's own follow-up work on data quality architecture for data warehouses makes a related point at the pipeline level [8], that quality has to be designed into the movement of data rather than checked for afterward, which is consistent with why the canonical mapping layer in Section IV-C sits ahead of the comparison engine rather than downstream of it.

The specific problem of testing a system or ETL migration has its own prior work. Sharma and Attar propose a generalized test framework for big data ETL migrations built around record counts, checksums, and column-level comparisons between source and target [9], which is the same family of technique the comparison engine in Section IV belongs to; the contribution here is narrower and more domain-specific, an insurance policy administration schema with a canonical mapping layer and a fixed refresh cadence tied to a live migration cutover, rather than a general-purpose framework. Commercial ETL and data quality testing tools, surveyed for example by Datagaps, generally take a similar record- and rule-based approach and can be configured to run against a full population rather than a sample [10]; what distinguishes the tool described here is less the comparison technique itself than where it lives, embedded directly in the same Power BI layer that already serves as the migration's day-to-day monitoring surface, so a QA analyst working a defect and a business stakeholder watching migration progress are looking at the same underlying fact table rather than two disconnected systems [11].

III. PROBLEM STATEMENT

The block of business covered by this validation tool includes more than 50,000 policy records, more than 150,000 coverage and rider records, and close to 500,000 role occurrences (the various parties attached to each policy). At the time of writing, the tool tracks more than 100 policy attributes, , which works out to roughly 3 million individual source-to-target attribute comparisons across the three record types combined. No manual review process can check a population of that size for every migration batch, and a small sample cannot reliably catch a defect that is scoped to a single field, a single role type, or a single plan code [12], because the odds of that specific combination showing up in a small sample are low.

A validation approach for this kind of migration therefore needs three properties: it has to check the full population, not a sample; it has to distinguish a defect that affects one record from a defect that affects an entire attribute or an entire class of role; and it has to run often enough, and stay current enough, to catch a regression introduced by a later migration batch rather than only the first one.

IV. VALIDATION TOOL ARCHITECTURE AND METHODOLOGY

A. EXTRACT ACQUISITION

Source data is queried directly from the source system's database and written out as source extracts, including a combined policy and coverage extract which are uploaded to a shared location. Target data is queried directly from the target system's database on its own server, producing separate extracts are loaded in Microsoft Azure DB. A migration error report, generated automatically at the time of each migration run, and a manually maintained policy exception report are brought in alongside the extracts. Figure 1 shows how these pieces come together.

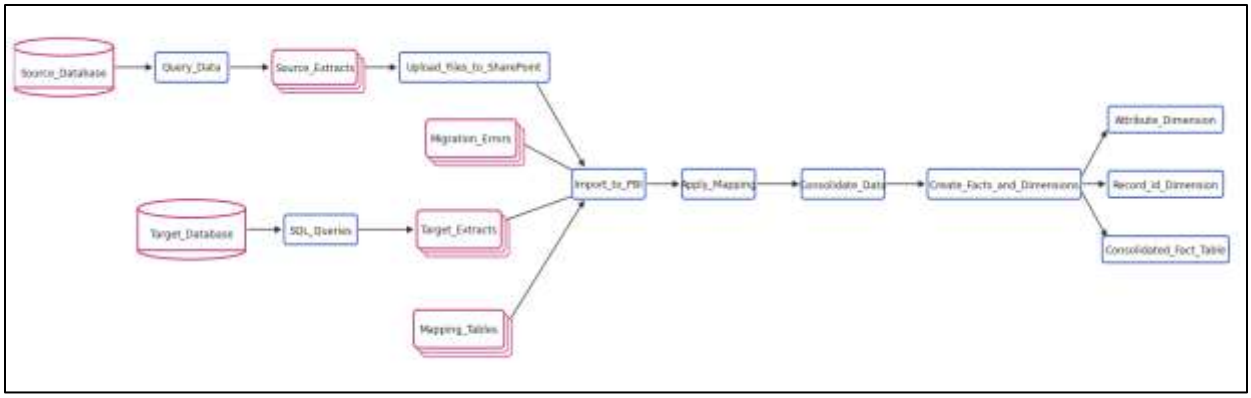

Figure 1. Data flow from the source and target databases, through source and target extraction, into the Power BI model. Migration error output and mapping tables are imported alongside the extracts; the mapping step, consolidation step, and fact/dimension build follow.

B. UNIQUE KEY DESIGN

Policy Number is the key used to match a policy record, and its associated coverage record, between the source and target extracts [13]. Role records need a different key, because a single policy has several role occupants, so the unique key for a role comparison is the combination of Policy Number, member identifier (renamed from the source's Member Number to match the target's Client_Id column), and Role Code [14]. Without the Role Code component, the tool could not tell apart, for example, the policy owner's date of birth from the primary beneficiary's date of birth on the same certificate [15].

C. CANONICAL TRANSFORMATION LAYER

Before any value is compared, both sides are normalized through mapping tables that translate each system's internal codes into a shared, canonical representation. Table 1 shows a representative subset of the mapping tables actually in use. Status_Code 1, for instance, is canonically “Inforce” on both sides; Dividend_Option 1 in the source system corresponds to code C in the target system, both meaning “paid in cash.” Without this layer, the comparison engine would flag every one of these pairs as a mismatch even when the underlying business meaning is identical, so the mapping tables are what make a literal, code-level comparison meaningful.

TABLE 1. Representative Canonical Transformation Rules

Attribute	Source Value	Target Value	Canonical Meaning
Status_Code	1	Inforce	Policy in force
Status_Code	3	Paid-up	Paid-up policy
Status_Code	B	Lapsed	Lapsed policy
UW_Class	STD	0	Standard risk class
UW_Class	TBC	3	Tobacco-use class
Dividend_Option	1	C	Paid in cash
Dividend_Option	3	A	Paid-up additions
Method_Code	4	E	EFT payment
Method_Code	1	D	Direct bill
Mode	12	A	Annual premium mode
Mode	1	M	Monthly premium mode
NFO	2	2	Reduced paid-up

D. COMPARISON ENGINE AND STATUS TAXONOMY

For every matched record and every attribute the tool tracks, the comparison engine assigns one of four statuses: Match, Mismatch, Missing Source Data, or Missing Target Data. A record that exists in the source but has no corresponding value in the target is Missing Target Data; the reverse is Missing Source Data [12]. This distinction turns out to matter more than a simple pass/fail count, because the two failure modes usually point

to different root causes. In the policy validation data captured for this project, for example, the Agent1_ID, Agent1_Split, Agent2_ID, Agent2_Split, Agent3_ID, and Agent3_Split fields for the certificates reviewed all came back Missing Target Data rather than Mismatch, meaning the agent-of-record fields were not populated in the target at all for those certificates, not populated incorrectly.

E. STAR-SCHEMA CONSOLIDATION

After the mapping step (Apply_Mapping), matched and compared records are consolidated (Consolidate_Data) and loaded into a small star schema (Create_Facts_and_Dimensions): a Consolidated_Fact_Table holding one row per record-attribute comparison, an Attribute_Dimension describing each tracked attribute, and a Record_Id_Dimension describing each certificate or role occurrence. This structure is what lets the four Power BI report pages (Dashboard, Policy Validation, Coverage Validation, Role Validation) slice the same underlying comparisons by attribute, by certificate number, by plan code, without re-running the comparison itself. The design follows standard star-schema guidance for analytical models: a fact table for the measurable events (record-attribute comparisons) surrounded by dimension tables describing the entity.

F. SCORING AND AGGREGATION

A Validation Score is computed as attributes passed divided by attributes tested, and it is rolled up at three levels: overall, per attribute, and per certificate or record. The overall score answers "how much of this migration reconciles," but on its own it can hide very different underlying problems, which is why the per-attribute and per-record breakdowns matter more in practice; Section VI works through why.

G. EXCEPTION MANAGEMENT

Known, already-ticketed defects are tracked by JIRA number and can be excluded through an exception filter in the report. With known issues filtered out, the remaining ("blank") records are expected to reconcile at 100%; any residual mismatch in that filtered view is, by definition, a defect that has not yet been triaged, which is what the QA team is meant to act on first.

H. OPERATIONAL CADENCE

The Power BI model refreshes on a fixed schedule tied to the migration batch cycle, four times a day at 6:00 AM, 9:00 AM, 12:00 PM, and 6:00 PM Central time. This cadence means a defect introduced in one day's migration run is visible to QA the same day, rather than waiting for a single overnight refresh.

I. IMPLEMENTATION: TYPE-SPECIFIC COMPARISON LOGIC

Not every attribute can be compared the same way. A certificate number is compared as text; a face amount is compared as a rounded currency value; an issue date has to be compared as a date, not as whatever text string each system happens to export [16, 17]. The patterns below are representative of the logic behind the Validation Status introduced in Section IV-D, implemented in two layers: Power Query M during extraction and staging (Section IV-A), and DAX calculated columns and measures in the star schema built in Section IV-E. They are shown here to document the approach, not reproduced verbatim from the production model.

1) Text: Text attributes such as Certificate_Number, Plan_Code, and beneficiary clause names are compared after trimming whitespace and normalizing case, so a value that differs only in spacing or capitalization is not flagged as a mismatch (Listing 1).

```
Text Status =
VAR SrcTxt = TRIM(UPPER(IF(ISBLANK(FactTable[Source_Value]), "", FactTable[Source_Value])))
VAR TgtTxt = TRIM(UPPER(IF(ISBLANK(FactTable[Target_Value]), "", FactTable[Target_Value])))
RETURN
  SWITCH(
    TRUE(),
    SrcTxt = "" && TgtTxt = "", BLANK(),
    SrcTxt = "" && TgtTxt <> "", "Missing Source Data",
    SrcTxt <> "" && TgtTxt = "", "Missing Target Data",
    SrcTxt = TgtTxt, "Match",
    "Mismatch"
  )
```

Listing 1. DAX measure computing Text Status for text-type attribute comparisons.

2) Numeric: Numeric attributes such as Issue_Age and Modal_Factor are converted from text to numbers and compared with a small tolerance rather than exact equality, since one system occasionally carries a decimal place the other has rounded away (Listing 2).

```
Numeric Status =
VAR SrcTxt = TRIM(FactTable[Source_Value])
```

```
VAR TgtTxt = TRIM(FactTable[Target_Value])
VAR SrcNum = IFERROR(VALUE(SrcTxt), BLANK())
VAR TgtNum = IFERROR(VALUE(TgtTxt), BLANK())
VAR Tolerance = 0.01
RETURN
  SWITCH(
    TRUE(),
    SrcTxt = "" && TgtTxt = "", BLANK(),
    (SrcTxt = "" || ISBLANK(SrcNum)) && TgtTxt <> "", "Missing Source Data",
    SrcTxt <> "" && (TgtTxt = "" || ISBLANK(TgtNum)), "Missing Target Data",
    ABS(SrcNum - TgtNum) <= Tolerance, "Match",
    "Mismatch"
  )
```

Listing 2. DAX measure computing Numeric Status with a tolerance-based comparison.

3) Date: Dates are the one type that cannot safely be parsed row by row inside a DAX measure, because the source and target systems do not export dates in the same text format; the Issue_Date values pulled from the source system came through as plain M/D/YYYY text, such as 9/1/1948. Parsing happens earlier, in the Power Query M step that builds the staged extract, using a function that tries a short list of known formats before giving up (Listing 3).

```
ParseDate = (value as any) as nullable date =>
  let
    txt = Text.Trim(Text.From(value ?? "")),
    result =
      if txt = "" then null
      else try Date.FromText(txt, [Format = "M/d/yyyy", Culture = "en-US"])
      otherwise try Date.FromText(txt, [Format = "yyyy-MM-dd"])
      otherwise null
  in
    result
```

Listing 3. Power Query M function parsing source and target date text into a comparable date type.

Once both sides carry a true Date type rather than text, the DAX comparison is a straightforward equality check (Listing 4).

```
Date Status =
VAR SrcDate = FactTable[Source_Date]
VAR TgtDate = FactTable[Target_Date]
RETURN
  SWITCH(
    TRUE(),
    ISBLANK(SrcDate) && ISBLANK(TgtDate), BLANK(),
    ISBLANK(SrcDate) && NOT ISBLANK(TgtDate), "Missing Source Data",
    NOT ISBLANK(SrcDate) && ISBLANK(TgtDate), "Missing Target Data",
    SrcDate = TgtDate, "Match",
    "Mismatch"
  )
```

Listing 4. DAX measure computing Date Status once both sides carry a parsed date type.

This is also the most likely explanation for the coverage attribute discussed in Section VI: if Issue_Date for coverage records was never routed through a parsing step like the one above, every row would reach the model as unparsed text rather than a comparable date, and every comparison for that attribute would come back Missing Target Data, consistent with the 0.0% coverage score reported in Table 3. This paper does not confirm that as the cause; it is offered as the kind of specific, checkable hypothesis this implementation pattern is meant to produce.

4) Currency: Currency attributes such as Modal_Premium, Face_Amount, and Loan_Balance arrive with dollar signs, thousands separators, and occasionally parentheses for negative values. These are cleaned to a plain number in Power Query M before they reach the model (Listing 5).

```
ParseCurrency = (value as any) as nullable number =>
```

```
  let
    txt = Text.Trim(Text.From(value ?? "")),
    cleaned = Text.Remove(txt, {"$", ",", " "}),
    isNegative = Text.StartsWith(cleaned, "(") and Text.EndsWith(cleaned, ")"),
    stripped = if isNegative then Text.Middle(cleaned, 1, Text.Length(cleaned) - 2) else cleaned,
    parsed = try Number.Round(Number.FromText(stripped), 2) otherwise null
  in
    if txt = "" then null
    else if isNegative and parsed <> null then -parsed
    else parsed
```

Listing 5. Power Query M function cleaning currency text into a signed numeric value.

The comparison itself follows the same shape as the numeric case, with the tolerance held to a cent (Listing 6).

```
Currency Status =
VAR SrcAmt = FactTable[Source_Amount]
VAR TgtAmt = FactTable[Target_Amount]
VAR Tolerance = 0.01
RETURN
  SWITCH(
    TRUE(),
    ISBLANK(SrcAmt) && ISBLANK(TgtAmt), BLANK(),
    ISBLANK(SrcAmt) && NOT ISBLANK(TgtAmt), "Missing Source Data",
    NOT ISBLANK(SrcAmt) && ISBLANK(TgtAmt), "Missing Target Data",
    ABS(SrcAmt - TgtAmt) <= Tolerance, "Match",
    "Mismatch"
  )
```

Listing 6. DAX measure computing Currency Status with a one-cent default tolerance.

5) Combining the four types: Attribute_Dimension carries a Data Type field, Text, Numeric, Date, or Currency, for each of the 46 policy, 15 coverage, and 4 role attributes tracked. One calculated column on the fact table reads that field and routes each row to the matching status column above (Listing 7).

```
Validation Status =
SWITCH(
  TRUE(),
  RELATED(Attribute_Dimension[Data Type]) = "Text", FactTable[Text Status],
  RELATED(Attribute_Dimension[Data Type]) = "Numeric", FactTable[Numeric Status],
  RELATED(Attribute_Dimension[Data Type]) = "Date", FactTable[Date Status],
  RELATED(Attribute_Dimension[Data Type]) = "Currency", FactTable[Currency Status],
  "Unclassified"
)
```

Listing 7. DAX calculated column routing each fact row to its type-specific status.

The Validation Score reported throughout Section VI is a measure over this column, counting only rows where at least one side carried a value (Listing 8).

```
Validation Score =
VAR Applicable = FILTER(FactTable, NOT ISBLANK(FactTable[Validation Status]))
VAR Passed = FILTER(Applicable, FactTable[Validation Status] = "Match")
RETURN
  DIVIDE(COUNTROWS(Passed), COUNTROWS(Applicable))
```

Listing 8. DAX measure computing the Validation Score reported throughout the paper.

Rows where both source and target are blank for a given attribute are excluded from the denominator rather than counted as a match, so an attribute that legitimately does not apply to a given record does not inflate its validation score.

6) Attribute-level tolerance: A single tolerance is not appropriate for every attribute. Face_Amount and Loan_Balance are carried directly from the source record and should match to the cent; a mismatch there is a real data problem. Modal_Premium, by contrast, is often the output of a rounding step, converting an annual premium into a monthly or quarterly amount and back, and the two systems do not always round that

conversion identically, so a strict one-cent tolerance flags premium rows as mismatches that are really just rounding artifacts of that calculation, not migration defects.

To handle this, Attribute_Dimension carries a Tolerance column set per attribute rather than a single constant buried inside a DAX measure. Table 2 shows representative settings: premium fields are given a wider allowance of $0.20, while identifiers, amounts carried straight through from the source, and role-level attributes stay at the tighter default.

TABLE 2. Representative Attribute Tolerance Settings

Attribute	Data Type	Tolerance
Modal_Premium	Currency	$0.20
Annual_Premium	Currency	$0.20
Face_Amount	Currency	$0.01
Loan_Balance	Currency	$0.01
Cost_Basis	Currency	$0.01
Dividend_Balance	Currency	$0.01
Modal_Factor	Numeric	0.001
Issue_Age	Numeric	0
(any unlisted attribute)	Numeric or Currency	$0.01 (default)

An attribute with no row in this table falls back to the default one-cent tolerance already shown in the Numeric Status and Currency Status logic, rather than failing or being skipped, so adding a newly tracked attribute never silently disables tolerance checking for it.

A match reached through tolerance is still recorded as Match in the Validation Status column and counts toward the Validation Score the same as an exact match, since the point of setting a tolerance is that the underlying values are considered equal for reconciliation purposes at that attribute's expected precision. The raw Source_Value and Target_Value are kept in the fact table regardless, so a reviewer working a specific certificate in Policy Validation or Coverage Validation can still see the actual dollar difference behind a tolerance-driven match; it is excluded from the Mismatch count, not hidden from the record-level detail.

V. CASE STUDY:

As part of migration policies are heavily impacted if Issue Age is not correctly migrated which will result all policy financial values [18]. We have noticed significant amount of policies are mismatched. This migration showed the Issue Age attribute reconciling at only 47.0% across all policies, and reviewing the certificates behind that number showed the shortfall was not spread evenly across policies. The gap was concentrated in calculating issue age logic is incorrect in target system.

Investigating directly in the target system's beneficiary filed some beneficiary designation record in the target system carries a Request Type field with exactly two valid values, Percentage or Monetary Amount, and a corresponding value field, either a percent share or a dollar amount [19, 20], depending on which type is selected. A correctly configured designation: Request Type is set to Percentage, and the value field reads 100% for a sole primary beneficiary, or 50% for each of two contingent beneficiaries. But defective designation on a different certificate: Request Type defaulted to Monetary Amount, and the value field reads $0.00.

In the source system, these beneficiary allocations were recorded as percentage splits, not dollar amounts. The migration's transformation logic had no rule translating a source percentage allocation into the target system's Request Type plus percentage-value pairing, so the target system's own default (Monetary Amount, $0.00) was left standing for every beneficiary designation that came through this path. The practical effect is that the beneficiary grid, the screen a service representative or the policyholder would look at to confirm who is entitled to what share of the death benefit, showed a $0.00 entitlement instead of the intended 50% or 100% share. For a whole life product where the beneficiary designation is the entire point of the record, this is not a cosmetic defect.

The fix, once the mapping gap was identified, was to add a transformation rule carrying the source's percentage allocation type and value into the target system's Request Type and percentage fields, re-run the affected certificates through the migration, and confirm the correction through the same tool, using the JIRA-linked exception filter to isolate the affected records until the fix was verified and the exception could be removed.

What made the defect visible in the first place was not a single record failing a spot check; it was that the tool compared the beneficiary role on every certificate, every refresh, so a rule that was silently wrong for an entire category of beneficiary records showed up as a clear, attribute-level pattern rather than a handful of unrelated complaints months after cutover.

VI. Results And Discussion

Table 3 summarizes the three validation reports as captured at the time of writing.
These are counts over the full record population rather than a sample, so no confidence interval applies in the usual statistical sense; the uncertainty that matters here is not sampling error but whether the canonical mapping and comparison logic themselves are correct, which Section VIII returns to.

TABLE 3. Validation Snapshot

Report	Records	Attributes	Attributes Tested	Attributes Passed	Validation Score
Policy Validation	50,000	56	2,800,000	1,437,623	51.3%
Coverage Validation	200,000	15	3,000,000	1,752,345	58.4%
Role Validation	50000	6	300,000	168,531	56.1%

None of these three numbers means the same thing, and reading them without the attribute-level breakdown behind each one would be misleading.
The policy score of 51.3% is a blend of attributes that reconcile almost completely (fifteen of the 56 tracked attributes were at or above 98%, including certificate number, issue date, and issue state) and a smaller group that reconcile poorly.
The coverage score of 58.4% is the starkest number in the tool, and it illustrates a different way a full-population comparison earns its keep.
This paper does not claim to have confirmed which of those is the actual cause; the point is that the tool's job here is not to explain the 0.0%, it is to make a defect of this scale impossible to miss, and to hand the QA team a specific, bounded question (why does this one attribute fail for every record) instead of an unbounded one.

VII. TESTING EFFORT SAVED

The project did not keep a formal log of QA hours before and after the tool was put in place, so the figures in this section are estimates, not measured results. They use deliberately fast, conservative assumptions about how long a manual comparison would take, and are meant to show the order of magnitude of effort involved, not a precise return-on-investment number. Section X outlines a timed-pilot design that would let a future iteration of this work replace these estimates with measured figures.
Table 4 estimates how long a manual, field-by-field comparison of the current record population would take, working from the same record and attribute counts reported in Table 3.

TABLE 4. Estimated Manual Comparison Effort at Current Record Counts

Record Type	Records	Attributes Tracked	Assumed Time / Record	Estimated Hours	Approx. 8-Hour Workdays
Policy	50,000	56	3 min	2500	312.5
Coverage	200,000	15	1.5 min	5,000	625
Role	50,000	6	1 min	833	260
Total	300,000	—	—	8,333	1042.5

Even at these fast assumed rates, three minutes for a 56-attribute policy record works out to under four seconds per attribute, and one minute for a role record works out to fifteen seconds per attribute, a single full-population pass comes to roughly 2,500 hours, or about 312 eight-hour workdays. For one analyst working alone, that is close to two years of full-time effort, and it is effort that would need to be repeated, not banked, on every subsequent migration batch, since a later batch can regress a field that an earlier one got right. The tool performs the same comparisons automatically inside each scheduled Power BI refresh, the four-times-daily cadence described in Section IV-H, at no added labor per run.The effort saved is not only in raw hours. What kept the Section V defect from being caught earlier was not that beneficiary records were scarce, but that

a manual review built around opening a handful of sampled policy screens does not routinely open the beneficiary designation sub-screen for every certificate it touches.

VIII. LIMITATIONS

The methodology described here has real boundaries. A canonical mapping table that is itself wrong, for example a status code mapped to the wrong target value, produces a confident Match on both sides even though the underlying business meaning has been changed; the tool checks consistency with the mapping rules, not correctness of the rules themselves. Role comparisons depend on a hand-maintained composite key and a hand-maintained role-code mapping, so a role type that has not yet been added to that mapping will not be compared at all rather than being flagged as missing. Because the model refreshes four times a day rather than continuously, a defect introduced between refreshes is not visible until the next scheduled run. Finally, the Validation Score is a useful summary but, as Section VI shows, it can average together very different underlying conditions, so any score reported at the top level should be read alongside its attribute- and record-level breakdown rather than on its own.

IX. Conclusion

A core system migration in insurance moves a large, business-critical population of records through transformation rules that are easy to get right for most fields and easy to get wrong for a specific field, role type, or record subset. Sampling-based review is not well suited to catching a defect of that shape. The methodology described in this paper, full-population, attribute-level comparison against a canonical mapping layer, refreshed on the same cadence as the migration batches and tracked against known issues by ticket number, caught a beneficiary allocation defect that would otherwise have reached a policyholder-facing screen, and it continues to surface both narrowly scoped and systemically scoped defects across the policy, coverage, and role populations described in Section VI. The approach is not specific to the particular source-to-target system pair described here in its mechanics, full-population comparison against a canonical mapping, type-specific comparison logic, and a refresh cadence tied to the batch cycle, and those mechanics are not tied to any one source or target platform. Whether it in fact generalizes to other policy administration system conversions, as opposed to merely appearing portable in principle, is not something this single-project account can establish; Section X outlines what would be needed to test that claim rather than assert it.

X. FUTURE WORK

Three extensions follow directly from the limitations noted in Section VIII and the estimate-only figures in Section VII, none of which have been carried out for this paper.

First, the effort-saved figures in Table 4 rest on assumed comparison rates rather than measured ones. The record-level and attribute-level comparison results this tool already produces would let a retrospective study draw a stratified random sample of certificates, time how long a manual reviewer actually takes to check the same attributes by hand, and compare that measured rate against the assumed rates used here, without needing to change the tool itself. This simulation has not been run; the data needed to run it already exists in the project's validation history, but analyzing it is future work.

Second, and more directly, a small timed pilot, a fixed set of certificates reviewed manually by QA staff under observation, alongside the same certificates run through the tool, would give a measured effort comparison in place of the conservative assumptions used in Section VII, and would also surface any defect classes the tool's current comparison logic misses that a human reviewer would catch.

Third, generalizing beyond the particular source-to-target system pair described here would mean applying the same canonical-mapping-plus-full-population-comparison approach to a different source or target system, or a different line of business, and checking whether the same class of narrowly scoped defect is caught there too. A related extension is automating part of the root cause classification in Section IV-D and IV-F, for example flagging when a Mismatch pattern is concentrated in one role type or plan code strongly enough to suggest a mapping error rather than a source-data problem, which is currently done by a QA analyst reading the per-attribute breakdown rather than by the tool itself.

CONFLICT OF INTEREST STATEMENT

The author declares no conflict of interest.

DATA AVAILABILITY STATEMENT

The record- and attribute-level validation data analyzed in this study contain policyholder and beneficiary information from a live insurance migration and are not publicly available due to privacy and contractual restrictions. Aggregated, de-identified summary figures are reported in Tables 2 through 4. The data analyzed in this study is confidential and are not publicly available due to privacy.

REFERENCES

[1] T. Moloi and A. F. Mulaba-Bafubiandi, "Management of disruptive technologies as applied in stages of long-term insurance processes," in *Proc. 2024 Portland Int. Conf. Management of Engineering and Technology (PICMET)*, Aug. 2024, pp. 1–16.

[2] R. Sommer, "Starting a fundamental transformation: From stone age to exploring the universe in a few years—Breaking the continuum of evolution in insurance," in *Transforming Public and Private Sector Organizations: Implementing Sustainable Purpose, Travelling Organization and Connectivity for Resilience*. Cham, Switzerland: Springer International Publishing, 2022, pp. 251–272.

[3] M. R. Sureddy and P. Yallamula, "Debugging methodology for issues in data warehousing and business intelligence platforms," *Int. J. Manage. Appl. Sci.*, vol. 6, no. 11, pp. 24–29, 2020.

[4] Ö. Gültekin, E. Cinar, K. Özkan, and A. Yazıcı, "Real-time fault detection and condition monitoring for industrial autonomous transfer vehicles utilizing edge artificial intelligence," *Sensors*, vol. 22, no. 9, Art. no. 3208, 2022.

[5] J. R. Machireddy, "Data quality management and performance optimization for enterprise-scale ETL pipelines in modern analytical ecosystems," *J. Data Sci., Predictive Analytics, Big Data Appl.*, vol. 8, no. 7, pp. 1–26, 2023.

[6] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proc. 46th Int. Symp. Computer Architecture*, Jun. 2019, pp. 541–553.

[7] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, 1996.

[8] M. R. Sureddy and P. Yallamula, "Data quality architecture for data warehouses," *Int. J. Res. Cultural Soc.*, vol. 4, no. 6, pp. 95–100, 2020.

[9] R. Sharma and V. Attar, "Generalized big data test framework for ETL migration," in *Proc. Int. Conf. Computing, Analytics and Security Trends (CAST)*, Pune, India, 2016, pp. 148–153, doi: 10.1109/CAST.2016.7915025.

[10] T. Alexakis, E. Adamopoulou, N. Peppes, E. Daskalakis, and G. Ntouskas, "Evaluating data quality: Comparative insights on standards, methodologies, and modern software tools," *Electronics*, vol. 14, no. 15, Art. no. 3038, 2025.

[11] C. C. Law, C. C. Chen, and B. J. Wu, "Managing the full ERP life-cycle: Considerations of maintenance and support requirements and IT governance practice as integral elements of the formula for successful ERP adoption," *Comput. Ind.*, vol. 61, no. 3, pp. 297–308, 2010.

[12] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Trans. Softw. Eng.*, vol. 50, no. 4, pp. 911–936, 2024.

[13] X. Wang, X. Qin, M. B. Hosseini, R. Slavin, T. D. Breaux, and J. Niu, "Guileak: Tracing privacy policy claims on user input data for Android applications," in *Proc. 40th Int. Conf. Software Engineering*, May 2018, pp. 37–47.

[14] N. Kshetri, "Blockchain's roles in meeting key supply chain management objectives," *Int. J. Inf. Manage.*, vol. 39, pp. 80–89, 2018.

[15] W. Hunter, "Identity documents, welfare enhancement, and group empowerment in the Global South," *J. Develop. Stud.*, vol. 55, no. 3, pp. 366–383, 2019.

[16] K. F. Widaman, "III. Missing data: What to do with or without them," *Monographs of the Society for Research in Child Development*, vol. 71, no. 3, pp. 42–64, 2006.

[17] N. K. Tyagi and M. Goyal, "Blockchain-based smart contract for issuance of country of origin certificate for Indian customs exports clearance," *Concurrency Comput.: Pract. Exper.*, vol. 35, no. 16, Art. no. e6249, 2023.

[18] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117782–117801, 2020.

[19] H. De Haas, M. Czaika, M. L. Flahaux, E. Mahendra, K. Natter, S. Vezzoli, and M. Villares-Varela, "International migration: Trends, determinants, and policy effects," *Population and Development Review*, vol. 45, no. 4, pp. 885–922, 2019.

[20] M. S. Farooq, M. Khan, and A. Abid, "A framework to make charity collection transparent and auditable using blockchain technology," *Comput. Electr. Eng.*, vol. 83, Art. no. 106588, 2020.