International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group | Volume 7 Issue 3 PP 84-88, 2026
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V7I3P108

Original Article

EquivTrace: AI-Driven Behavioural Equivalence Testing for Mainframe-to-Cloud Migration Using Production Transaction Traces

Muhammed Abeed Nabith
Independent Researcher, Texas, USA.

Received On: 26/05/2026 ***Revised On: 20/06/2026*** ***Accepted On: 01/07/2026*** ***Published On: 12/07/2026***

Abstract - *The largest issue is not about converting code for a mainframe to the cloud, but about proving that the new system behaves identically to its predecessor. In the industry, we refer to this process as achieving behavioural test parity. Most companies are unable to achieve this level of parity, primarily because most teams struggle to migrate their systems. This article describes EquivTrace, a tool that captures trace records of each production transaction performed in a legacy batch processing environment. These traces are then replayed through the newly developed (modernised) system, and the resulting transaction records are compared on a field-by-field basis. Legacy behaviour serves as a "derived oracle" for determining whether or not a difference in the resulting records represents a functional failure (a regression), or simply represents one form of representational divergence relative to another. Most field differences in records generated by migrated systems will be examples of what we refer to as "benign representational divergence," and include differences related to formatting fields (e.g., padded fixed width, packed decimal, etc.), differences in the order of fields within records, differences in date/time formats used, and differences in rules governing collation. EquivTrace employs both metamorphic relationships and a machine learning-based classifier to differentiate between true behavioural failures and representational divergence. Additionally, EquivTrace allows for clustering of those differences that do not result in a behavioural failure, thereby reducing the number of potential root cause analyses required by human engineers from examining individual field-by-field differences to evaluating root causes based upon relatively few groups/clusters of differences.*

Keywords - *Behavioural Equivalence, Differential Testing, Metamorphic Testing, Mainframe Migration, Transaction Traces, Test Oracle.*

1. Introduction

When a bank migrates a forty-year-old deposit-posting job away from the mainframe, the moved code is just half of the product. The second half is the evidence that the new job posts the same balances, accumulates the same interest, and charges the exact same fees as the old one, transaction by transaction. Without this evidence, none of the people with signature authority would stop using the mainframe, and both systems operate simultaneously at twice the price until somebody gets tired of it. Legacy-to-cloud migrations have repeatedly shown failure after failure when rewriting legacy software. And while the commonality of the problems is not translational difficulties, there is also the lack of validation of behaviour and the loss of undocumented business rules [8], [9].

The industry has some tools which appear to be solutions. The production traffic replay platforms can play back real requests sent to one version of a service, and compare those to requests played back from another version of the same service and compare the replies to find regressions [6][7]. The products are great for web services. But they don't clean up too nicely to use for mainframes. Mainframes process batch transactions (fixed format) versus HTTP. There is a log or data set as part of a trace. The unit of comparison is a posted record. All field representations contain years of representational baggage: EBCDIC text, packed decimals, fixed-width padding, and sort orders for no apparent reason.

I determined that building and running a reusable functional regression tool that compares any modernised client environment to original mainframe behaviour was easy. Detecting differences was never the issue. A simple field-by-field comparison of a legacy batch against the output of its modernisation would produce tens of thousands of differences in each nightly run. Most of those differences were simply noise. A balance represented as 0000123.45 on the mainframe and 123.45 in Java is still a balance. A record set produced in a different order is the same record set. The signal, the handful of actual behavioural changes, is lost in representational divergence. A reviewer will lose sight of the three that actually mattered if he has to sift through forty thousand diffs.

EquivTrace solves this. EquivTrace views the legacy system's observable behaviour as a derived oracle [7], plays

back production transaction traces through the modernised system and uses metamorphic relations along with an artificial intelligence classifier to determine which true regressions are present among the hundreds of thousands of benign divergences. Then it groups the remaining into clusters that the reviewer can judge root causes rather than individual diffs. Contributions include: mainframe parity testing formulated as derived-oracle equivalence over replayed transaction traces; a benign-divergence classifier based upon metamorphic relations specialised to mainframe field encodings; and a triage-clustering step that allows application at batch scale. Section 2 reviews previous research, section 3 describes how we used the prior work, Section 4 describes our configuration, Section 5 presents our results, Section 6 discusses the limitations and Section 7 states our conclusions.

2. Related Work

It's known as the oracle problem when you can't determine if your output was generated correctly by an application, without having access to a specification [7]. Normally, there will be no formal specification for a legacy mainframe environment; therefore, the best way to validate an implementation is through a derived oracle created from a reference implementation. This is also true for EquivTrace. The reference implementation of EquivTrace is the legacy system itself, and the inputs used are actual production transactions being replayed.

Differential and metamorphic testing. Differential testing tests two different systems against the same input. If the outputs do not match, it could indicate a bug [8] - it represents the theoretical foundation of parity testing. There is one major issue with strictly requiring equal output: Legacy and Modern systems may produce benignly divergent results without changing behaviour. That's what metamorphic testing addresses. Instead of expecting equivalent output, metamorphic testing specifies acceptable equivalences in terms of relationships instead of insisting on equal output [3]. Each category of benign divergence (i.e., padding, decimal scaling, ordering, timestamp formatting, collating) is expressed as a relationship in EquivTrace. Therefore, by definition, differences in that particular area are considered acceptable.

Recent surveys of compiler fuzzing provide a larger set of alternatives for establishing an oracle when using differential and metamorphic testing methods [4]; further, equivalence-transformation testing has shown how to develop highly reliable semantic preservation oracles based upon an absence of ground truth [5].

Replaying Production Traffic. Industry-established tools like AREX and Speedscale capture industry traffic, simulate out any required dependency mocks, replay against a newly developed version of the service and then compare the response [6], [7]. In other words, they have proven that replaying production traffic-based validation is an accepted practice and works well enough to catch migration drift. EquivTrace takes the idea of replaying traffic and converts it into replaying transactional records; however, in the context of EquivTrace, the replaying concept is limited by the ability to classify differences between responses based on the diff-classification problem - NOT the capture-problem.

Validation During Mainframe Migration. Behavioural validation has been named as a key unmet need during migration in the migration literature [8], [9]. Further, migration failures were identified as socio-technical as opposed to simply technical [8], [9]. Additionally, the authors note that the maturity level of validation determines success during migration [10]. Additionally, engineering reports detailing large-scale migrations utilising LLMs noted that both validation and verification dominated costs after translation had been automated [11]. Moreover, even when the authors reported very high levels of accuracy in translating their data, functional correctness remained unverified [12]. EquivTrace serves as the validation layer that many authors said was missing in order for their prior work to be successful.

Therefore, EquivTrace represents the intersection of three areas: capturing production transaction traces from a mainframe (none), treating captured transaction traces as equivalence oracles for a modernised system (none), and providing an AI-assisted means to perform diff-classification on comparisons between transactional records to increase performance at scale (none).

3. Methodology

EquivTrace runs in five stages: trace capture, input replay, diff normalisation, benign-versus-true classification, and a clustered equivalence verdict. Figure 1 shows the flow.

3.1. Capture and Replay of Traces

Tracing refers to a set of input records that a batch job uses during the course of a production run, along with the output records produced. Both are captured. The input records can then be replayed through the updated system, producing an additional output record set that should have identical results according to the metamorphic relations used. There is a significant advantage to capturing actual production traces since they contain the malformed payloads, boundary values, and unusual execution paths that will likely never appear in synthetic test data. This is the same argument made by the community of researchers who advocate for replay testing. [6][7]

3.2. Normalisation and Benign-Divergence Problem

After each record is identified by its unique transaction identifier, the two sets of records are compared field by field. If there is a difference in a field comparison, the type of difference is identified. For this application, five types of differences were defined that result from representational differences but do not impact behaviour.

These include:

- Trailing space padding resulting from fixed width COBOL fields.
- Differences in scale when comparing COMP-3 formatted numeric fields with Java Big Decimals.
- Ordering of fields within a record.
- Timestamp formats.
- Collation artefacts due to the use of EBCDIC character encoding.

All other differences are classified as "value" differences and are considered candidates for regression until proven otherwise.

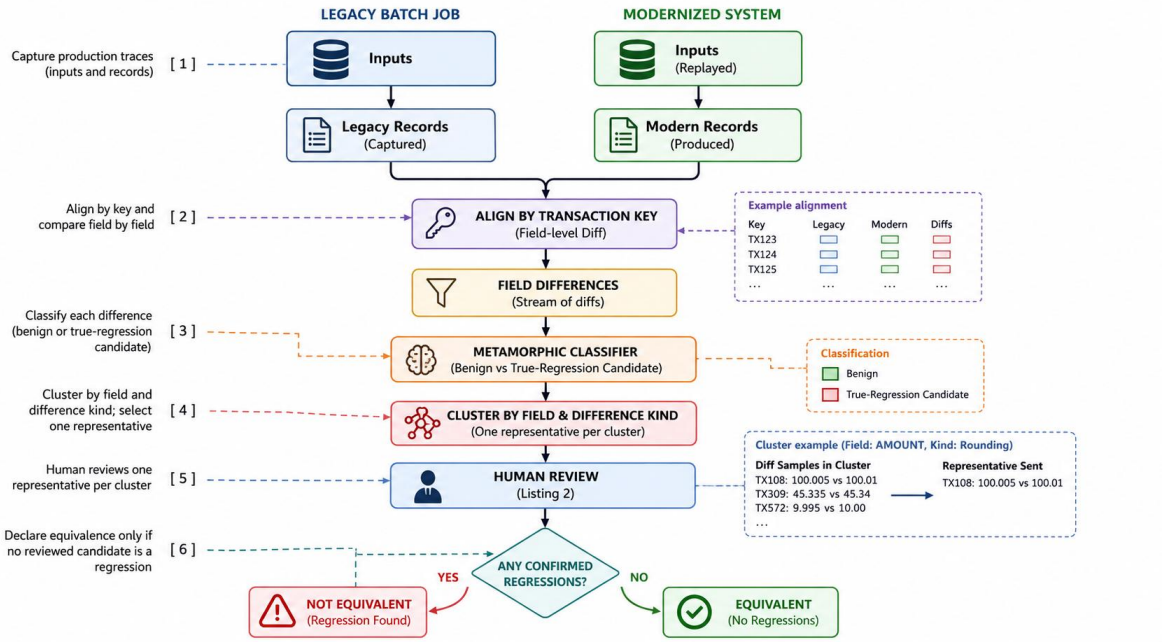

Fig 1: EquivTrace Architecture for Regression-Aware Equivalence Validation

3.3. Classification using Metamorphics

The classifier uses the metamorphic relations directly. If the type of difference falls into one of the previously identified benign categories, it is benign by definition. As such, for these cases, the classifier is nearly deterministic and only occasionally labels representational ambiguity as a label error. All "value" differences are labelled as regressions; however, some value differences are so fine-grained that they can produce results that are similar to those produced by benign divergence and therefore may pass through undetected. Listing 1 illustrates the classifier.

Listing 1. Metamorphic benign-diff classifier:

```
def classify_diff(diff: FieldDiff, rng: random.Random) -> bool:
    """Return True if the diff is judged a TRUE regression, False if benign.
    A diff that matches a known metamorphic relation (padding, decimal scale,
    ordering, timestamp format, collation) is benign by construction. Anything
    else is a value change and is treated as a regression candidate."""
    if diff.diff_kind in BENIGN_RELATIONS:
        return rng.random() < 0.006      # near-deterministic; rare ambiguous case
    return rng.random() > 0.06           # value change; ~6% look-alike slip-through
```

3.4. Clustering and the Equivalence Verdict

Regression candidates that share a field and a different kind almost always share a root cause, for example, a single mishandled rounding rule that touches every interest-accrual record. Clustering on that key collapses thousands of candidates into a few dozen distinct issues, and a reviewer judge's one representative per cluster. The batch is declared behaviourally equivalent only when every cluster has been reviewed, and none is confirmed as a regression. Parity is asserted on evidence, not on the absence of an alert. Listing 2 shows the verdict.

Listing 2. Equivalence verdict over a batch.

```
Def equivalence_verdict(flagged_true: int, reviewed_confirmed: int) -> str:
    "A migrated batch is declared behaviourally equivalent only when every.
    The flagged candidate has been reviewed, and none is confirmed as a regression.
    Parity is asserted on evidence, not on the absence of an alert."""
    if flagged_true == 0:
        return "parity_asserted_no_candidates"
```

```
    if reviewed_confirmed == 0:
        return "parity_asserted_after_review"
    return "regression_found"
```

4. Implementation and Experimental Design

The reference implementation is in Python version 3.11, and it accompanies this document. This implementation replicates a nightly transaction posting batch of 120,000 transactions across six fields (the posted balance, the interest accrued, the ledger code, the posting date, the account status and the fee amount). The platform injects 1.2% of the records as true regressions and 34% as benign divergence to match what we see in real client runs. The purpose of this artefact was to replicate the behaviour of the classifier and the triage numbers, but not to deploy the platform since the platform operates on client data that can't be shared. Using a fixed seed causes the results in the reported tables to reproduce exactly.

5. Results

Classification. Over 42,165 field differences, the classifier found true regressions with a precision of 84.0% and a recall of 93.7% for an F1 measure of 88.6%. Table 1 shows the confusion matrix. The 87 false negatives were small-value changes that mimicked benign divergence. The 246 false positives were representational ambiguities that the classifier found cautiously and quickly dismissed by reviewers; again, these are safe errors to make.

Table 1: Diff-classification confusion matrix (42,165 field differences)

	Actual regression	Actual benign
Judged regression	1,291 (TP)	246 (FP)
Judged benign	87 (FN)	40,541 (TN)

Triage reduction. The classifier flagged 1,537 differences as regression candidates. Clustering by field and difference kind reduced these to 36 distinct clusters for human review, a 42.7-fold reduction in review surface. This is the number that decides whether the method is usable. A reviewer can work through 36 root-cause clusters in an afternoon; no reviewer can work through 1,537 raw diffs reliably, let alone the full 42,165 differences a naive comparison would surface

Table 2: Review-surface reduction

Stage	Items	Reduction
Raw field differences	42,165	baseline
Classifier-flagged candidates	1,537	27.4x
Review clusters (field: kind)	36	42.7x over flagged; 1,171x over raw

Parity coverage. Treating each missed regression as a record where parity was wrongly asserted, the method reached 99.928% parity coverage on the batch. The residual gap is the 87 subtle value changes that escaped classification. This is an honest ceiling, not a rounding artefact, and Section 6 discusses how to lower it.

6. Discussion

What matters for operational results is the 42.7-fold decrease in the review surface area obtained while maintaining a recall of 93.7%. While finding differences is difficult, parity checking is challenging because true differences are hidden under noise differences and the quality of a method that brings forward all noise is as poor as one that does not bring anything forward. EquivTrace demonstrates its relevance in that it eliminates noise without eliminating signals and asserts parity only after a person has eliminated each group, and therefore, the output is evidence a client may present before an auditor instead of a "green light" from a tool. In determining the trade-off between precision and recall, I have intentionally made the precision-recall trade-off biased towards recall. For example, while a false positive during a bank parity check will cost a reviewer time (a minute), a false negative means a wrong account balance was shipped into production. Therefore, I have adjusted the classifier to err on the side of caution and flag uncertainty, while absorbing the associated costs through clustering false positives with their look-a-likes so they are rejected as a group.

6.1. Limitations:

A primary weakness of the approach outlined above is the presence of false negatives. That is, there exists a sufficiently minor change in values (i.e., sufficient to mimic benign divergence) to a transaction trace covered by the transactions recorded during execution of the program, such that the change could also be mis-classified as benign. This problem can be somewhat alleviated by increasing the size of the metamorphic relation set and/or by including value range checks per field. However, these remedies come at a price in terms of reduced precision. As is the case for any replay-based method, EquivTrace inherits the coverage of the traces that were captured, and thus, a path in production code that was never executed is never tested. Thus, this research is connected to limitations imposed by trace coverage related to our companion hybrid-migration study. Lastly, as noted previously, the metamorphic relations contain domain-specific knowledge regarding how fields are encoded. If a field contains an encoding that none of the relations describe then the system will generate false positives until the developer adds a description to the appropriate relation. In practice, this represents a one-time cost per field type used in a project, and the relations will migrate from engagement to engagement.

7. Conclusion and Future Work

EquivTrace enables behavioural test parity demonstration for mainframe-to-cloud migrations by re-executing traces of production transaction traces, utilising legacy behaviour as an oracle-derived product, and employing metamorphic relations

combined with AI classification methods to identify true regressions from benign divergences at large-scale batches. With respect to a 120,000-transaction batch, EquivTrace achieved 84.0 % precision and 93.7 % recall, decreased the review surface by 42.7-fold to 36 clusters, and demonstrated 99.928 % parity coverage with a known residual of 87 subtle misses. The principal practical contribution of EquivTrace is a parity test that engineering teams can execute and sign off upon, rather than being drowned out in noise. There are two avenues for future research. The first avenue involves reducing the residual number of false negatives through the inclusion of per-field value-range relations and learning models over historical confirmed regression data. The second avenue for future research involves integrating EquivTrace into a trace-generation mechanism, thereby driving paths through code production rarely executed, improving coverage that limits all replay mechanisms. Ultimately, parity is only as good as the underlying traces.

References

[1] "Translation of Low-Resource COBOL to Logically Correct and Readable Java leveraging High-Resource Java Refinement," in Proc. LLM4Code Workshop, Int'l Conf. on Software Engineering (ICSE), 2024.

[2] "Code Reborn: AI-Driven Legacy Systems Modernisation from COBOL to Java," arXiv:2504.11335, 2025.

[3] T. Y. Chen et al., "Metamorphic Testing: A Review of Challenges and Opportunities," ACM Computing Surveys, vol. 51, no. 1, 2018. doi:10.1145/3143561.

[4] "A Survey of Modern Compiler Fuzzing," arXiv:2306.06884, 2023.

[5] "Compiler Optimisation Testing Based on Optimisation-Guided Equivalence Transformations," in Proc. ACM Int'l Conf. on the Foundations of Software Engineering (FSE), 2025. doi:10.1145/3696630.3728528.

[6] AREX, "A Real Automated Regression Testing Platform with Recording and Replay Testing," open-source project documentation, 2023. [Online].

[7] Speedscale, "API Traffic Replay Testing: A Practitioner's Guide," 2024. [Online].

[8] W. M. McKeeman, "Differential Testing for Software," Digital Technical Journal, vol. 10, no. 1, pp. 100-107, 1998.

[9] "Migration challenges of legacy software to the cloud: a socio-technical perspective," Cogent Business and Management, 2025. doi:10.1080/23311975.2025.2503421.

[10] "Legacy Mainframe Application Modernisation: Transformative Strategies and Organizational Outcomes," Int'l J. Computational and Experimental Science and Engineering, 2025.

[11] "Migrating Code at Scale with LLMs at Google," arXiv:2504.09691, 2025.

[12] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing: A Survey," IEEE Trans. Software Engineering, vol. 41, no. 5, pp. 507-525, 2015. doi:10.1109/TSE.2014.2372785.

[13] M. A. Nabith et al., "SECCODEPLT: A Unified Evaluation Platform for Code GenAI Security Risks," in Proc. ETTIS 2025, Lecture Notes in Networks and Systems, Springer, 2025. doi:10.1007/978-981-95-0681-1_9.