Change the rule in the documentation, not the code: full report of a pre-registered test of agent-built COBOL migrations against the legacy program

FlowX.AI

RECAST full report · Version 1.1, 2026-11-17, DOI 10.5281/zenodo.24305235 (Version 1.0, 2026-11-03, DOI 10.5281/zenodo.24223975, is unchanged) · FlowX.AI · Apache-2.0 · Repository: https://github.com/flowx-ai/carddemo-equivalence-study. Every number below traces to a file in the repository; the source is given in brackets, and §8 explains the paths.

Abstract.

We ask how closely an agent-built FlowX.AI implementation of a legacy batch step reproduces the legacy program's behavior, when the only description the builder receives is a wiki that another agent generated from the source; whether the build says in advance where it may differ; and whether a business-rule change made in the wiki alone is carried into the rebuilt program. The subject is the nightly batch chain of AWS CardDemo, open source (Apache-2.0): interest calculation (INTCALC, CBACT04C), transaction posting (POSTTRAN, CBTRN02C), statements (CREASTMT, CBSTM03A/B) and the daily transaction report (TRANREPT, CBTRN03C). The oracle is the COBOL compiled with GnuCOBOL 3.2, unmodified apart from one patch that removes diagnostic DISPLAYs, run with a pinned clock on the shipped data and on four evaluation sets derived from the programs' branches. The protocol, scorer and golden outputs were frozen and their hash anchored in the Bitcoin blockchain before any build was scored; every later experiment was added by a dated amendment, committed before its builds and in most cases also timestamped before them (§3.1). Two builders, the FlowX Developer Agentic Harness (the in-product AI builder) and FlowX Platform MCP (a coding agent working through the platform's MCP server), build every batch step, three independent builds per condition, and every output record is scored against the oracle. The two builders are reported separately and never pooled; the Agentic Harness results are in §4.9.

Change request. We edited one business rule in the wiki only and compared each rebuild with the legacy program carrying the same edit. CR2, the headline, moves POSTTRAN's over-limit check to the credit limit × 1.10, a change for which we know of no public source (we kept no search log for it). CR1, the contrast, rounds INTCALC interest to the cent, a change CardDemo's public tracker already proposes. All six FlowX Platform MCP builds followed the change: CR2 scores 100% on every set, CR1 98.99–99.03% with the legacy last-account record as its only divergence; the unedited legacy program scores 91.1–92.3% against CR2 and 43.4–44.7% against CR1. H11 is supported for FlowX Platform MCP on both rules. On the Agentic Harness, all six builds put the edited rule in their code: CR1 equals FlowX Platform MCP on every set in 3 of 3 builds (H11h supported), and CR2 matches the edited legacy on every record in 1 of 3 (H11h not refuted for CR2, with the per-build criterion met by one build). Six further rules, one or two per step and none found proposed in public, were pre-registered under A21. No FlowX Platform MCP build kept a legacy rule: 16 of 18 followed the change (H14 not refuted), and the other two differ on an unrelated, flagged address field. On the Agentic Harness, 18 builds count toward the same six rules: none kept a legacy rule silently, and H14h is not refuted on each rule and overall. CR3, CR5 and CR7 followed the change in 3 of 3 builds (CR7 at 99.36–99.37%), CR4 in 1 of 3. CR6 and CR8 are "other" in 6 of 6: every statement differs from legacy on a flagged address line, so they score 0% at record level, and only a post-hoc field-level count shows the changed rule followed.

Ablation under disguise. On the disguised INTCALC, FlowX Platform MCP builds from the detailed wiki came within 1 point of builds given the disguised COBOL (0.996 points on the pre-registered median; on one set of five the gap is 0.01 over the limit), so H12 is supported at the margin. A summary-only wiki scored as well as the detailed wiki in business terms (H13 not supported), and builds with no wiki scored

about 1–4% (the instrument check passes, though these builds lean on recall of CardDemo). The Agentic Harness, whose ablation builds showed no sign of recall of CardDemo, supports H12 at the same margin and also H13: without the business rules section its builds lose 50 points, where FlowX Platform MCP filled the gap from memory.

FlowX Platform MCP. From the detailed wiki, all twelve FlowX Platform MCP builds return output on every set: 98.99–99.03% of records on INTCALC, 100% on POSTTRAN, 100% of statements on at least four of five CREASTMT sets, and 99.72–99.73% of TRANREPT report units. What remains is a legacy defect the builds did not copy (the last INTCALC account is never rewritten; the TRANREPT grand total counts the last amount twice) or a choice at a card on which legacy abends; all but one build flagged these before scoring. From the thin wiki, matches come largely from memory of the public CardDemo source: a disguise control turned 99–100% on INTCALC into about 1%.

Agentic Harness. In the pre-registered first scored run, 7 of 24 Agentic Harness builds returned output. The primary condition returned no records, so H1, H2 and H4 are not supported and H3 is not assessable there. On POSTTRAN, CREASTMT and TRANREPT no Agentic Harness build that returned output differed from legacy without a flag. Two exploratory re-runs on later versions are reported beside it and never pooled with it: A15 (19 of the 24 builds; 12 returned output as first decoded, 14 with two labelled post-hoc decoding corrections) and A18 (a) (all 24 builds; output in all 24, and 7 with a silent divergence at base) (§4.9).

Two lessons follow. When the documentation omits a rule, the score measures the model's prior knowledge of the system, which a private codebase will not offer. And part of the honesty measured on public code is memory: in the main CardDemo arms one FlowX Platform MCP build that returned output differed from legacy silently, while under the disguise four of six did. The last-account divergence is also silent in five FlowX Platform MCP change-request builds.

Screens. A registered test (A24) built four online screens with FlowX Platform MCP from one wiki and scored them against an online oracle that runs the unmodified COBOL on a CICS stub we wrote. H15, the payment's file state, is not refuted. H16–H19 (screens, account-view data, browser subset, end to end) are refuted, each by 3 of 3 builds, so the target claim A24 registered is not stated (§6). In the Agentic Harness arm of the same test (A26–A31) no screen was built: H15h–H18h are refuted on two of three counted builds, both scored absent, and H19h was not run (§4.9). Of the three registered builds, the two counted ones (A30 r1, A31 r2) produced no screens and are scored as absent under the registered rule; if those two builds are set aside, no verdict is reached.

Full application. A32 extends the test to the whole base application, FlowX Platform MCP only: 17 online programs and 11 batch programs, scored on 54 screen cases, 9 browser cases, 23 batch cases and 4 end-to-end pairs, three builds at $35.62–37.67 each. HF3 (the money files) holds in 3 of 3 builds; HF1, HF2 and HF4–HF7 are each refuted 3 of 3, so the target claim, batch to working screens, both checked against the original, is not stated (§6.3). The Agentic Harness is not part of A32.

Screen specification and confirmatory test. We then added a screen specification derived from the COBOL source (17 program pages, 8 batch pages, a catalogue of 150 legacy defects) and build rules, developed over a series of exploratory builds in which only these inputs changed, and froze them. On the development run, the last build before the freeze (exploratory, FlowX Platform MCP, $67.96, 2 h 11 min), all 54 screen cases equal legacy on untagged units (3,147 of 3,147); tagged units are 4,624 of 4,627, the three differences resting on two instrument limits fixed in advance; all 23 batch cases and the 4 core steps on five sets match; 4 of 4 end-to-end pairs carry through; the browser check passes 60 of 60 steps. Because the inputs were developed on these cases, this is not evidence of equivalence. The confirmatory test, pre-registered and timestamped before the first build, ran three fresh builds on the frozen inputs (FlowX Platform MCP, $48.88–61.21 and 1 h 37 min–2 h 06 min each). Two pass all five conditions: every scored unit equals legacy (38,298 of 38,298) apart from 29 units excused by two catalogued defects the build

flagged and 2 under an instrument limit fixed in advance; the browser check passes 60 of 60 steps, all batch cases and end-to-end pairs match. The third passes four conditions and fails online on 2 units of one screen: on the card list, a second PF8 on the last page shows the first press's message, a catalogued legacy behaviour (LD-62) the build stated it reproduced. The registered claim needed 3 of 3 builds, so full equivalence is not claimed (§6.5).

Summary for non-specialist readers

What we did. We took AWS CardDemo, open source (Apache-2.0): Amazon's sample of a credit-card mainframe application, and its four nightly COBOL batch programs. One AI read the COBOL and wrote documentation; a second AI, given only that documentation, rebuilt each program on the FlowX.AI platform. We ran old and new on the same data and compared every output record. The registered predictions and scoring rules were fixed before the builds they govern were scored; later changes are labeled.

The headline: change a business rule by editing the documentation. We changed one rule in the documentation, and made the same change in the old COBOL to know the expected output. The rule is that a purchase is over the credit limit only beyond the limit plus 10%. We know of no public source that proposes it; we kept no search log for this rule.

- The FlowX Platform MCP rebuild matched the patched legacy system on every record, in 3 of 3 independent builds.
- A second change, rounding interest to the cent, matched 99% in 3 of 3 of the FlowX Platform MCP builds. The one record per test set that differs is a defect the legacy program already has, not the change.
- Six more rule changes, across all four programs: FlowX Platform MCP followed the change in 16 of 18 builds and kept the old rule in none. The other two differ on an unrelated address line they had flagged in advance.
- The protocol also registered a second builder, the Agentic Harness, reported separately and never pooled (§4.9). It put the headline rule in its code in 3 of 3 builds and matched every record in 1; it followed the rounding change in 3 of 3; and of its 18 builds of the six further rules, 10 followed the change and none kept an old rule without a warning.

Disguised, the documentation came within one record of the source code. CardDemo is public, so an AI may remember it. We renamed everything in the interest program and changed some values. Rebuilt from the documentation alone, the FlowX Platform MCP builds came within one record per test set of its rebuilds from the disguised code. With no documentation, they fell back on memory, which no longer held under the disguise, and scored about 1–4%. The in-product builder's arm of the same test is reported separately (§4.9).

Cost. A rebuild of one program by FlowX Platform MCP typically cost $2–5 in AI usage (median $3.46) and took 5–14 minutes (median 7.5) [§4.10].

Beyond batch. We ran a registered test of four of the application's screens (sign-on, menu, account view and bill payment): FlowX Platform MCP built them three times, and each build was scored against the original screens, run on a stand-in for the mainframe's online system. The bill payment's file changes matched on the two registered payment cases in all three builds; a third payment case, on a stand-in rule, differs in two builds, unflagged in one. The screens differ from the original: the registered tests of screen content, account-view data, browser behaviour and the end-to-end chain each failed in all three builds. So the result is that screens were built and scored against the original, and §6 shows how they differ. The Agentic Harness built no screen in its arm of the test, so its screens predictions are refuted as registered; that verdict rests on two builds scored as absent, and without them no verdict is reached (§4.9).

The whole application. We then registered a test of the rest of the base application: all 17 online programs and 11 batch programs, built three times by FlowX Platform MCP from one 234-page wiki, at $35.62–37.67 and 69–75 minutes per build. Of seven predictions, one held in all three builds: the screens that move money (bill payment, transaction add, account update) wrote the same file records as the legacy program, with no unflagged difference. The other six failed in all three builds: not every screen was reachable, at most one screen program per build had no unflagged difference, list paging, the browser check, the chain from screens into batch, and the export and import round trip. So "batch to working screens, both checked against the original" is not claimed (§6.3).

The whole application again, with a screen specification. We then wrote a screen specification derived from the COBOL source and a set of build rules, and developed both over a series of development builds before freezing them. Every change between those builds was to these written inputs; no built application was edited by hand. On the last development build, which is exploratory, the screens matched the legacy run on all 54 screen cases on the units that rest on no stub rule, and on 4,624 of 4,627 of the units that do; every batch case matched, all 4 chains from the screens into batch carried through, and the browser check passed 60 of 60 steps. The inputs were developed against these same cases, so this result does not establish equivalence. The confirmatory test, pre-registered and timestamped before the first build, then ran three fresh builds on the frozen inputs. Two passed every condition. The third matched on everything except two units of one screen: on the card list, a second press of PF8 on the last page showed the message of the first press, a catalogued legacy behaviour the build said it reproduced. The test required 3 of 3, so full equivalence is not claimed (§6.5).

What we learned.

- Where the documentation leaves a rule out, an AI fills the gap from memory of public code; a private system offers no such memory, so documentation quality matters more there.
- Most rebuilds that differed from the legacy system said so in advance, in a written list of their assumptions; under the disguise, fewer did.
- On the whole application, what moved the screens toward the legacy run was a better written description, not repairs to a build: the inputs changed between development builds, and each build started fresh from them (exploratory, §6.4).

What we do not claim.

- The scale is small: four programs of 649–1,154 lines, 50 accounts, five test data sets, three builds per condition.
- CardDemo is a sample application, not a bank's production system, and it ran on GnuCOBOL, not on a mainframe.
- That screens built this way are equivalent to the original. The registered screens test failed on four of its five hypotheses for FlowX Platform MCP, and the full-application test on six of its seven. The development run that matched is exploratory and was tuned on the same cases; in the confirmatory test two of three fresh builds passed and one did not, so full equivalence is not claimed (§6.5). The reference for the online system is our own reading of the vendor's documentation, not a real one.
- We measure whether the rebuild behaves like the legacy program, not whether either is correct.
- This is FlowX.AI testing its own platform; the data, scorer and steps to rerun it are in the public repository, https://github.com/flowx-ai/carddemo-equivalence-study.

1 Introduction

Agentic tools now produce two artifacts in a legacy modernization: documentation recovered from the source, and an implementation built from that documentation. Both are usually judged by reading them. Reading cannot tell whether the new system computes what the old one did, record for record, and it cannot tell whether the tool knew where it was unsure.

We test that directly on a small, public, runnable subject. The legacy batch program runs off-mainframe and gives golden outputs; the agent-built implementation runs on the same inputs; a scorer compares every output record. We ask three questions, fixed before any build was scored (PROTOCOL §1):

- **RQ1, fidelity.** Starting only from the generated wiki, how much of the step's behavior does the build reproduce, with no human edits to the generated logic?
- **RQ2, honesty.** Of the records where it differs from legacy, how many did the pipeline flag as uncertain beforehand (*flagged*), and how many differ without warning (*silent*)?
- **RQ3.** How do the two builders and the two wikis differ?

The measure throughout is **equivalence with the legacy outputs, not correctness**. Where the legacy program has a defect, a build that fixes it differs from legacy. Whether a migration should keep a legacy defect is a business decision; the study measures whether the build matches legacy, and whether it said so in advance where it does not.

The batch part of this report covers four steps: the two of the pre-registered design (PROTOCOL §7), INTCALC first, then POSTTRAN under amendment A11, and the layer 1 steps CREASTMT and TRANREPT under A12–A14. Every step after INTCALC is secondary, reported per step, never pooled, and carries no hypothesis of its own (§4.7, §4.8). Two further parts each have their own amendment, timestamped before their builds: a change request (A16, hypothesis H11, and six further rules under A21, hypothesis H14; §4.1) and a wiki ablation under disguise (A17, hypotheses H12 and H13; §4.2). The second registered builder, the Agentic Harness, is reported in one place, §4.9, never pooled with FlowX Platform MCP. An unregistered pilot that rebuilt four online screens from the wikis is reported apart, in §6, and is not part of any result, finding or hypothesis; a registered test of those screens (A24, hypotheses H15–H19) follows it in §6.2, the full base application (A32) in §6.3, an exploratory development run with a source-derived screen specification in §6.4, and the confirmatory test in §6.5. The change request leads the results: it is the question a business asks of a migration, and the public code holds the old rule, so memory of it does not supply the new one. For the Agentic Harness, too, the change request is the headline result.

1.1 Related work

[paper/LITERATURE.md]

COBOL migration and equivalence testing. Testing a COBOL migration against the original is established. Hans et al.[1] turn symbolic execution of COBOL into JUnit tests for COBOL-to-Java transformations (FSE 2025 Industry); Froimovich et al.[2] combine analytic checkers with an LLM judge (2025); Ferenczi et al.[3] run COBOL and Java side by side off-mainframe with parity checks (2026); SEDCoT uses[4] GnuCOBOL as the reference for differential testing on 319 CodeNet programs with perturbed inputs (2026). AgentModernize uses[5] a behavioral specification graph as the intermediate artifact on eight synthetic scenarios (2026), and Alves et al.[6] report 70% equivalence and per-feature cost for a VB6 to C# migration with coding agents (2026). Our study differs in what the builder is given: no source code, only documentation another agent generated from it, and a pre-registered score against an independently cross-checked legacy run. GnuCOBOL as an oracle, differential testing against it, and feeding execution differences back to the translator (Pan et al., ICSE 2024[7]; UniTrans, FSE 2024[8]; TransAgent, FSE 2026[9]) are not contributions of this paper.

Change requests. Dao et al.[10] benchmark change-aware round trips between a data model and a schema (JSS 2026) and Hill studies[11] specification artifacts in open-source pull-request workflows (ESEM 2026); neither scores a change made in documentation against the legacy program re-run with the same change.

Documentation as a specification. Round-trip correctness (Allamanis et al., ICML 2024[12]), Code2Text2-Code[13], Documentary[14], SWD-Bench[15], Beyond Lexical Similarity[16], SpecTra[17], CodeWiki and[18] PROOF evaluate[19] generated documentation or specifications by regenerating code from them, usually against unit tests. None of these, in the searches made, score documentation-only rebuilds against an executed legacy program with a control for training-data recall.

Warnings, calibration and silent failures. Articulate but Wrong reports[20] that 31.7% of behavior drifts were silently endorsed by the model that produced them (2026); Confident and Wrong[21], Spiess et al.[22] on calibration (ICSE 2025) and HumanEvalComm measure[23] related properties after the fact. We know of no design that freezes the builder's own warnings before scoring and checks them against an executed oracle, as RQ2 does.

Recall and contamination. When Names Disappear[24] (ClassEval-Obf, 2025) and The SWE-Bench Illusion[25] (2025) show that public benchmarks partly measure memorization; our disguise arm (A5) applies the same concern to a public legacy codebase.

Pre-registration. Registered reports are established in software engineering (Ernst and Baldassarre[26]) and were the subject of NeurIPS workshops in 2020 and 2021[27]. Blockchain timestamping of protocols was proposed in 2016 in a paper since retracted (Irving and Holden[28]). We use timestamps for credibility; the package is published with this report, and the timestamps, not the publication date, carry the order.

1.2 What is and is not new here

- **Claimed:** documentation-only rebuilds scored against an executed, cross-checked legacy run under a pre-registered protocol; the flagged-versus-silent measure of whether a build announced its divergences before scoring; a recall control on a public legacy codebase; a change request made in the wiki only and scored against the legacy program with the same edit, with one change memory can supply and one it cannot; and a wiki ablation run under disguise. We found no prior work verifying a change request against the re-run modified legacy program in 3 searches, none on a memory-controlled documentation ablation against an executed legacy oracle in 3 searches, and no design that freezes warnings before scoring (`paper/LITERATURE.md`); we do not claim to be first.
- **Not claimed:** GnuCOBOL as an oracle, differential testing or fuzzing against it, execution feedback to the builder, round-trip documentation tests, or pre-registration as such.
- **Legacy defects were named, not discovered.** Several behaviors the builds named before scoring are already in CardDemo's public tracker: the last-account defect (issue #64)[29], the end-of-file double count in CBTRN03C and the missing `ROUNDED` in CBACT04C (pull request #32)[30], and the blank account group (issue #21)[31]. In five searches of the tracker we found nothing on the CREASTMT city truncation, the orphan-card abend, the 51×10 statement overflow or the null filler bytes.
- **The flagged-in-advance record depends partly on recall.** Among the FlowX Platform MCP builds of the main CardDemo arms that returned output, one build's divergence was silent (INTCALC FlowX Platform MCP Sonnet r1). Under the A5 disguise, four of six builds had a silent divergence (Sonnet-wiki r1 and r2 on the last account, Opus-wiki r2 and r3), and Sonnet-wiki r2 went from flagged to silent [`runs/A5_SUMMARY.md`]. Part of the honesty measured on public code comes from memory of it. The same last-account divergence is also silent in all three FlowX Platform MCP CR1 builds and in FlowX Platform MCP CR3 r2 and r3 (§4.1).

2 Subject and oracle

Subject. AWS CardDemo, open source (Apache-2.0), commit[32] `59cc6c2f`. Its batch chain is POSTTRAN, INTCALC, COMBTRAN, CREASTMT and TRANREPT. INTCALC (CBACT04C) reads the category balances left by POSTTRAN, looks up a rate per account group, transaction type and category, writes one interest transaction per rated category, and adds the interest to each account while resetting its cycle totals. POSTTRAN (CBTRN02C) reads the day's transactions, validates each one and either posts it (transaction file, account balance and cycle totals, category balance) or writes it to a reject file with a reason: 100 card not in the cross-reference, 101 account not found, 102 over the credit limit, 103 received after the account's expiration date. CREASTMT (CBSTM03A with its subroutine CBSTM03B) reads the combined transactions, cross-references, customers and accounts and prints one statement per card, as text and as HTML: name, address, account summary, credit score, and the card's transactions with their total. TRANREPT (CBTRN03C) prints the daily transaction report for a date window: one detail line per transaction, a total per account and per page, and a grand total. COMBTRAN, between INTCALC and CREASTMT, is a merge and a key sort with no COBOL program; it has no build arm (A12).

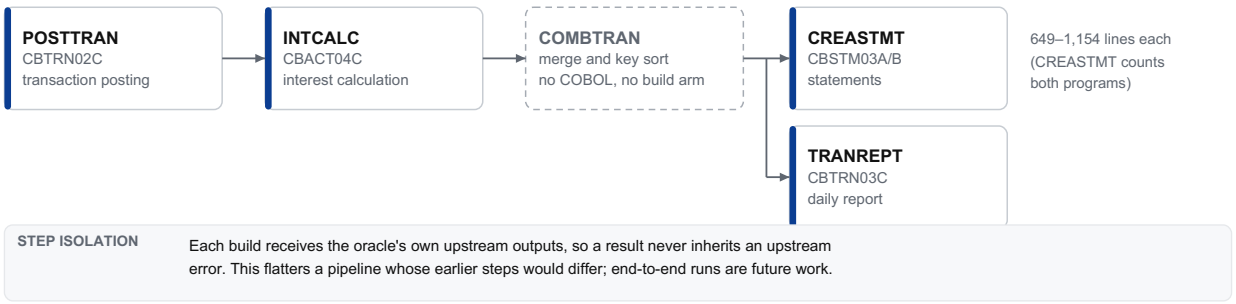

Figure 1. The batch chain of AWS CardDemo, open source (Apache-2.0). COMBTRAN is a merge and a key sort with no COBOL program, and it has no build arm. Each step is scored in isolation. Diagram: no data plotted.

Oracle. The COBOL is compiled unmodified with GnuCOBOL 3.2[33] (`-std=ibm -fsign=EBCDIC -ftab-width=3`). The only code we wrote are two off-mainframe shims (an abend stub and a PARM driver) and one patch to the statement program CBSTM03A [notebook §4, §9]. That patch comments out source lines 266–291, which walk z/OS control blocks from address 0 only to DISPLAY the job and DD names; they do no file I/O and set no field the program reads later. It matters only for CREASTMT, the one step in this study that runs CBSTM03A. The clock is pinned to hundredths and time zone; two runs are byte-identical [notebook §8]. Each step was also re-implemented independently in Python from the source, and both implementations agree on every record [notebook §6]. For CREASTMT and TRANREPT the independent cross-checks (`tools/ref_creastmt.py`, `tools/ref_tranrept.py`), written from the source and the JCL, match the GnuCOBOL outputs line for line on all five sets (10 of 10, A12). The golden outputs are frozen with hashes in `golden/v1`.

Six properties of the legacy steps matter for what follows.

1. *The last account is never rewritten.* CBACT04C's main loop sets end-of-file inside the branch that would otherwise reach the final account update, so the update never runs. Account 00000000050 receives its interest transaction (18.77) but its balance stays 1945.87 and its cycle totals are not reset. A statement trace confirms 49 account updates for 50 accounts [notebook §7, §17]. Neither generated wiki mentions this [notebook §13].
2. *Every account's group field is blank.* In the shipped account file the group code `A000000000` sits in the ZIP field; the group field the program reads is blank for all 50 accounts. Every rate lookup misses and falls

back to the `DEFAULT` group (CBACT04C lines 436–438). The numbers come out right only because `DEFAULT` and `A000000000` carry the same rates for every type and category present [notebook §11]. The defect in the data is latent; the fallback hides it.

3. *POSTTRAN's credit check uses the cycle totals, and a later reason overwrites an earlier one.* The legacy over-limit test is `credit limit >= cycle credit - cycle debit + amount` (CBTRN02C lines 403–410), not the current balance; the expiration test that follows runs regardless, so a transaction failing both is rejected with 103 (lines 414–418).
4. *CREASTMT keeps only the first word of the city.* The statement's third address line is built with `STRING … DELIMITED BY ' '` (CBSTM03A line 472), so a city of several words prints as its first word (a line reads "New OR USA 04257"). This affects 24 of the 50 customers, in every set [`golden/v1`].
5. *CREASTMT stops at a card with no account.* On `c03` the orphan cross-reference record makes CBSTM03A abend after it has written 44 statements; the statements written before the abend are kept (S5, A12).
6. *TRANREPT counts the last transaction twice and omits the last account total.* At end of file CBTRN03C adds the last amount to the totals a second time (lines 196–204) and never prints the total for the last card (lines 183, 306): the grand total on `base` is 79,254.29 where the detail lines sum to 79,233.86 (+20.43), and the report holds 49 account totals for 50 cards. Both hold in every set.

A further CREASTMT property is excluded by design: CBSTM03A holds each card's transactions in a 51 × 10 table with no bounds check, and `c04` gives account 00000000005 eleven transactions on one card. GnuCOBOL runs on where z/OS may abend (not verified), so that statement (text and HTML, 2 units) is excluded from `c04` equivalence and reported separately (S4, A12).

Evaluation sets [PROTOCOL §3; notebook §12, §18]. `base` is the shipped data. `c01` covers posting validation (POSTTRAN). `c02` covers the rate paths: own group found, zero rate (no interest transaction), unknown group (falls back). `c03` adds an orphan card cross-reference to a nonexistent account. `c04` adds the value partitions a wiki-independent branch inventory found uncovered: negative interest (the legacy truncates toward zero), several rated categories on one account, and an amount of 0. Across the five sets the two INTCALC outputs hold 502 records; the four POSTTRAN outputs hold 2,265 (posted / rejected transactions: base 262/38, c01 265/42, c02 262/38, c03 262/39, c04 266/38; notebook §40). The layer 1 scorer counts report units (§3.5): CREASTMT has 100 statement units on `base`, `c01` and `c02`, 88 on `c03` (the abend) and 98 on `c04` (S4), 486 in all; TRANREPT has 362–370 units per set, 1,824 in all, and legacy prints 18 pages per report [`runs/layer1/*_scores.md`].

Step isolation. Each INTCALC build receives the oracle's own POSTTRAN outputs as input, so an INTCALC result never inherits an upstream error (PROTOCOL §7). This flatters a pipeline whose earlier steps would differ; end-to-end runs are future work. POSTTRAN is the first step, so each set's own input files are its inputs. A CREASTMT build receives the oracle's combined transactions (`combtran_transact`) and INTCALC account file for the set, plus the set's own cross-references and customers; a TRANREPT build receives the combined transactions plus the set's cross-references, transaction types and categories (A13, A14).

3 Method

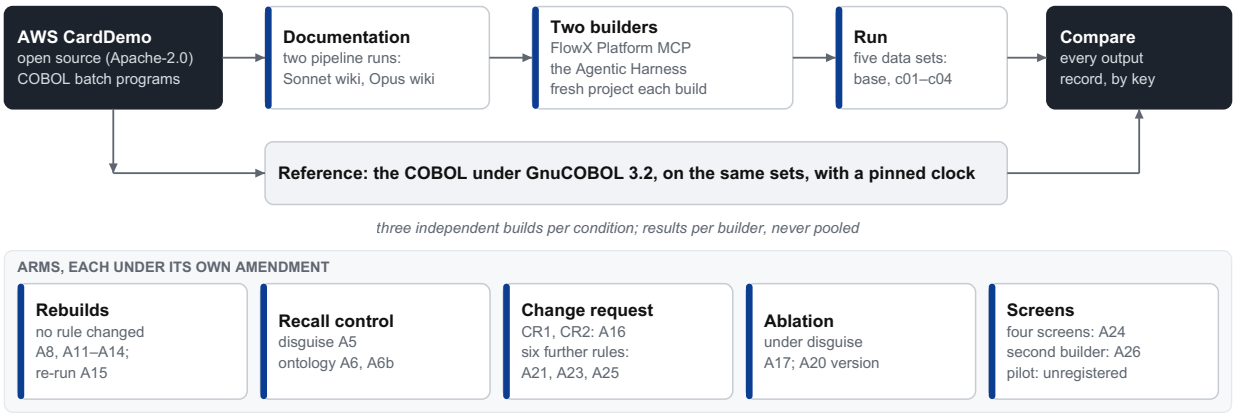

Figure 2. Study design. One AI documents the COBOL, a second builds only from that documentation, and every output record is compared with the COBOL run on the same data. Diagram: no data plotted.

3.1 Pre-registration

The protocol (`PROTOCOL.md`), the scorer, the golden outputs and the sources of the build bundles were frozen as git tag `protocol-v1` in the private study repository on 2026-10-24, before any build was scored. The Opus bundle's PDF was rendered and hashed on 2026-10-25 (private repository commit b156f64b), and editorial note E1 replaced two internal names in A2–A4. The pre-registration package is published with this report (`prereg/frozen/CardDemo_preregistration.zip`); its SHA-256 was stamped with OpenTimestamps[34] and is attested in Bitcoin block 973059 (block time 2026-10-25T10:44:06Z) [notebook §23–24]. Every later change is a dated amendment (A1–A32, PROTOCOL §13; A33, the confirmatory test, frozen and timestamped on 16 November 2026 before its first build); results are reported against the version they were produced under. Amendments written after a score was seen are labeled exploratory. The layer 1 amendments A12–A14 were written after every POSTTRAN build was scored and before any layer 1 build; a frozen copy (sha256 `cb3a9945…10e5c1`) was stamped the same way and is attested in Bitcoin block 973237 (block time 2026-10-26T12:41:33Z), before any layer 1 score. Bitcoin Core is not installed, so `ots verify` could not run locally; the proof's merkle root was checked against that block's header from a public block explorer [notebook §44–45]. The amendments for the Agentic Harness re-run (A15), the change request and ablation (A16–A18) and the Agentic Harness arms of A19 were each frozen and stamped the same way before any of their builds; at the time, only the hash was published. A15's earliest Bitcoin attestation is block 973299. The A16–A18 copy (sha256 `75e12d80…01bc00d`, committed with its stamp by 21:45:43Z on 2026-10-27) and the A19 copy (sha256 `350d86fe…5ac78`, stamped 2026-10-28) were submitted to four OpenTimestamps calendars [notebook §54, §59]. A20 (Agentic Harness arms), A21 (six further change requests, with a manifest of their golden outputs, models and bundles) and A22 (a retry rule) were frozen and stamped the same way on 2026-10-28, before any of their builds [notebook §61, §63, §66]. A23 (the Agentic Harness arm of the six further change requests) was frozen and stamped the same way on 2026-10-29, before any of its builds [PROTOCOL A23]. So were A24 (the screens test, sha256 `ca5f7d61…7b46`, 2026-10-29), A25 (the rest of the change requests, `873082d6…294d`, 2026-10-29) and A26 (the Agentic Harness arm of the screens test and an Agentic Harness re-run, `59f515bb…b429`, 2026-10-30) [`prereg/frozen/`]. So were A27–A31 (Agentic Harness arms, all 2026-10-30) and A32 (the full base application, FlowX Platform MCP only, protocol `e6971f58…3396`, 2026-11-02) [`prereg/frozen/`]. The earliest Bitcoin attestations of these stamps are blocks 973463 (A16–A18, block time 2026-10-27T21:50:26Z), 973480 (A19, 2026-10-28T01:41:55Z), 973528 (A20, 08:57:27Z), 973533 (A21, 09:56:03Z), 973540 (A22, 11:03:31Z), 973691 (A23, 2026-10-29T12:18:49Z),

973733 (A24, 20:22:47Z), 973738 (A25, 20:48:14Z) and 973793 (A26, 2026-10-30T08:22:55Z). Bitcoin Core is not installed, so we checked the merkle roots in the proofs against block headers from blockstream.info; all match [`protocol/VERIFY.md`]. The A27–A32 stamps are attested, by earliest height, in blocks 973807 (A27, 2026-10-30T10:30:58Z), 973824 (A28, 13:14:25Z), 973857 (A29, 18:09:18Z), 973863 (A30, 18:59:01Z), 973887 (A31, 22:23:20Z) and 974238 (A32, 2026-11-02T09:32:33Z), checked the same way [`protocol/VERIFY.md`].

Which amendments were stamped before their builds, and which rest on git commits:

- **Stamped before their builds:** A5 and A6 (block 973071, 2026-10-25T12:13:36Z), A7 (block 973082, 14:39:11Z), A12–A14, A15, A16–A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31 and A32. Each stamp was committed before the first build it governs.
- **In the package:** A1–A4 were committed before the first build (2026-10-25T09:37:18Z). The package's stamp (committed 10:08:01Z) and its block (10:44:06Z) came after that build began and before the first score (11:23:13Z).
- **Git commits only:** A6b, A8, A9, A10 and A11 have no stamp of their own; the A12–A14 stamp first covers them. For A11 that stamp came after every build it governs was scored. A9 and A10 are exploratory.

A block is an upper bound on a stamp's time; where it falls after the first build, the order rests on the git commit and stamp submission, our own records. That holds for the package and for A12–A14, A19, A20, A23, A24, A25, A27, A28, A30, A31 and A32; for A19 and A24 a build was also scored before the block. It holds in effect for A7, whose block time is 12 seconds before its first build (14:39:23Z), too close for block times to settle. For A5 and A6, A15, A16–A18, A21, A22 and A26 the block comes before the first build. For A29 it comes after its unscored smoke build and 41 seconds before its first build.

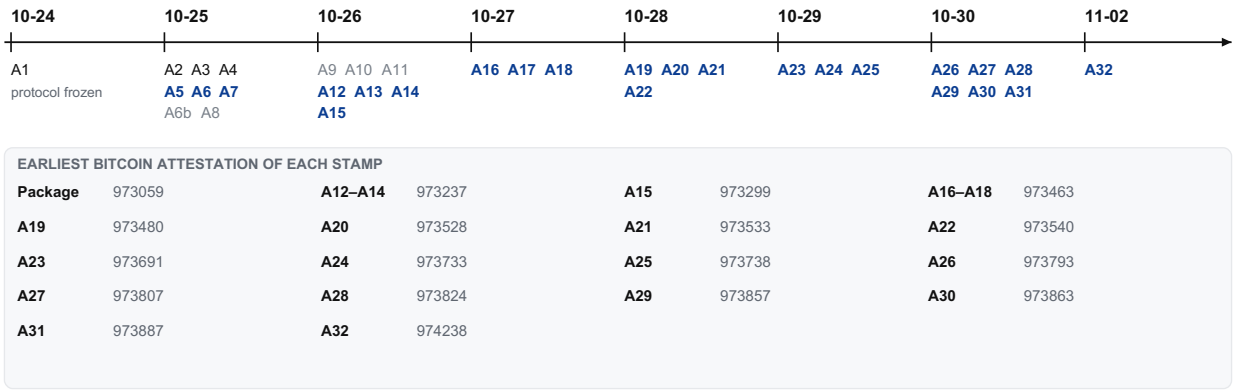

Figure 3. Amendment timeline, A1–A32, by the date in PROTOCOL.md; A25 is dated 2026-10-29/30. Bold: committed and stamped before the builds it governs. Black: in the package, whose stamp and block came after the first build began and before the first score. Grey (A6b, A8–A11): git commits only, first covered by the A12–A14 stamp. A block is an upper bound on a stamp's time; for the package and for A12–A14, A19, A20, A23, A24, A25, A27, A28, A30, A31 and A32 it falls after the first build (§3.1). Dates, status and block numbers are parsed from the protocol, this report and the study's records.

3.2 The wikis and the build bundles

Both wikis were generated from the CardDemo source by an agentic documentation pipeline, in two separate runs whose models and agent versions differ. We call them the *Sonnet wiki* and the *Opus wiki* after the model each run used. The names refer to the wiki, never to the builder. Because the runs differ in more than the model, any comparison between the wikis is observational [PROTOCOL §10].

A mechanical rule selects each bundle: the step's process page, its acceptance-criteria page, the data model, every page the process page links to, and the output contract (PROTOCOL Appendix A). The Sonnet bundle has 7 pages plus the contract (25 PDF pages); the Opus bundle has 3 pages plus the contract (11 PDF pages). For POSTTRAN the same rule gives 8 pages plus the contract (28 PDF pages) and 4 pages plus the contract (13 PDF pages); pages shared with the INTCALC bundles are byte-identical (A11).

A close read of the two interest pages [notebook §13] found:

Table 1. What the two wikis state about the interest step, from a close read of the two interest pages.

Legacy behavior	Sonnet wiki	Opus wiki
Formula: category balance × rate ÷ 1200	stated	different formula (account balance × rate), marked TODO
Truncation (no rounding)	stated, marked TODO	not stated (generic TODO)
DEFAULT-group fallback	stated	not mentioned
Zero rate → no transaction	stated	not mentioned
Last account never rewritten	not mentioned; the flow updates every account	not mentioned
Group field blank in the data	not mentioned; data model lists group values as if accounts carried them	not mentioned

For posting, the Sonnet wiki states every validation and posting rule; its own decision notes mark the order of the 102 and 103 checks as uncertain. The Opus wiki gives a different credit rule (`amount > credit limit − current balance`, plus active-status checks) and leaves the reject codes as a TODO ("see CBTRN01-03C source for exact reject reasons") [notebook §41].

For layer 1 the same rule gives, for CREASTMT, 8 pages plus the contract (29 PDF pages) from the Sonnet wiki and 5 plus the contract (16) from the Opus wiki; for TRANREPT, 8 plus the contract (29) and 6 plus the contract (19) (A13, A14). The Opus wiki has no page for the batch report; its step page is the reports process page, whose legacy source names the TRANREPT job and which also covers the online report request (CORPT00C). Each layer 1 contract adds `scorer/CONTRACT-L1.md`: the input files, the run parameters and the output line layouts (field positions, fixed labels, the HTML lines), with no assembly rule (A12). Neither wiki states any of the legacy properties 4–6. The Sonnet statement pages list the address as "3 lines, state, zip" with no rule for building line 3, and mark the statement line layout as a TODO; the Opus pages defer the layout and totals to a TODO ("Read CBSTM03A/B … for the exact statement layout and totals"). For the report, the Sonnet wiki lists page, card and grand totals with a TODO to confirm them, and leaves the page size a TODO ("lines per page in CBTRN03C"); the Opus pages state neither.

3.3 Builders

- **Agentic Harness.** The FlowX Developer Agentic Harness, the in-product builder, on a development environment (platform 5.13), driven through its API with the calls the Designer panel makes (A8). Every gate is answered mechanically: the option the Agentic Harness recommends, else the first; a fixed reply to any free-text question; brief and plan approved unchanged (A2). The model is whatever the platform resolves; it was `gpt-6.1-sol` in every run. Later Agentic Harness arms ran on later versions, each named in an amendment timestamped before its builds (A15, A18, A19, A20, A22, A23, A25, A26 and A27–A31), and results are reported per version, never pooled. The bundles, prompt, gate rules, adapters, scorers and flag procedure did not change between versions.

- **FlowX Platform MCP.** A fresh headless Claude Code session (`claude-opus-5-5`) with only the FlowX Platform MCP server and file tools limited to its own run directory (no shell, no web), building into a pre-created empty project (A3, A4). A headless session cannot ask questions; it decides alone. The session also inherits the operator's global instruction file, which the study did not intend as an input (§7).

Both builders receive the same PDF and the same prompt, verbatim (PROTOCOL Appendix A). The prompt asks the builder to record every decision it made where the wiki is uncertain or silent. Builders never see the golden outputs, the cases, the notebook or the protocol.

Design. Builder × wiki, three independent builds per condition, each in a fresh project. The pre-registered primary condition is Agentic Harness × Sonnet wiki. Allowed human interventions are infrastructure only (sign-in, deployment, retrying a platform call that did not complete, at most twice); anything else would make the run *assisted* (none was). A build that produces no output is scored as it stands; no run is rerun or dropped after its score is seen.

Run order. Amendment A3 ran the FlowX Platform MCP conditions first, and the Agentic Harness conditions ran under A8. The primary condition did not change.

3.4 Execution

Each build is run once per evaluation set through the platform's run endpoint. An adapter encodes the set's inputs in the contract's field names and decodes the returned lists into contract JSON. If a build names its fields differently, the operator may write a pure renaming map for that build, committed in its run directory; nothing else (A2, A4). A run that ends with no records leaves that set's outputs absent. For POSTTRAN the adapter has two modes, matching what the build accepts: record lists (daily transactions, accounts, card cross-references, category balances, and card records when a build asks for them) or the contract's fixed-width files as text, plus the business date (A11). The layer 1 adapters have the same two modes; record mode sends every copybook field and never the FILLER, file mode sends the files as the legacy step reads them. TRANREPT takes the report window 2022-07-01 to 2022-07-31 as run parameters (S3). Each adapter's self-test (encode, run the independent model, decode, score) gives 100% strict and business on all five sets in both modes (A13, A14; notebook §44). Every layer 1 build was run in file mode.

3.5 Scoring

- **Unit:** one output record, matched by business key (accounts by account ID; interest transactions by card and occurrence, since their IDs are a generated counter). Amounts compare as decimals; padding is ignored; record order is ignored. The run-clock timestamps are masked (PROTOCOL §4).
- **Equivalence** = equal records ÷ (golden keys ∪ candidate keys). A missing or extra record counts as unequal. *Business* equivalence is the primary endpoint; *strict* equivalence also compares presentation fields (descriptions, source). In every INTCALC and every POSTTRAN build the two coincide.
- **Rule tags.** Every divergence carries one tag from a closed catalog, assigned mechanically (PROTOCOL §5): for INTCALC, R-INT-RATE (amount off by more than 0.01, or a transaction missing or extra), R-INT-TRUNC (off by exactly 0.01), R-INT-APPLY and R-INT-CYC (balance or cycle totals after interest), R-INT-LAST (the last account), R-INT-TXFORMAT, R-PASSTHRU and R-ABSENT (an output not produced). For POSTTRAN: R-POST-100 to -103 (rejected for that reason in one run and not the other), R-POST-PRECEDENCE (102 and 103 swapped), R-POST-WRITE, R-POST-COPY, R-POST-BAL (balance or cycle totals after posting) and R-POST-TCAT (category balance).
- **Flags.** A flag is an explicit statement of uncertainty in a frozen artifact before scoring: a wiki TODO, or an assumption the build itself recorded. Before a build's outputs are opened, the flag mapper reads its assumptions and writes `flags.json`, each entry mapped to a rule and citing its source line; the file is

committed with a hash. A divergence under a flagged rule counts as flagged, otherwise silent. **The flag mapper is the AI assistant that ran the experiment (Claude).** The rubric is in notebook §25.

- **Instrument validation.** Before any build was scored, golden scored against itself gave 100% on all five sets, and a self-test of 13 planted defects returned the exact expected rule counts [notebook §14, §18]. Rounding half-up instead of truncating changes 28 of 50 interest amounts on `base`, so the rounding probe is not marginal. For POSTTRAN, an independent Python model of CBTRN02C agrees with the golden outputs on all five sets, and the adapter's self-test (encode, post with that model, decode, score) gives 100% strict on all five [notebook §40].
- **Layer 1 scorer** (`scorer/report_compare.py`, frozen at private repository commit 79d5e60; A12). Units: one statement per account in each statement output (text and HTML); for the report, each detail line (keyed by transaction ID), each account total, the grand total and the header. *Strict*: the unit's lines are identical after trailing spaces. *Business*: the unit's parsed fields are equal, with whitespace collapsed, low-values read as spaces, amounts compared as numbers and a statement's transactions as a multiset. Equivalence is computed per output as above; the two are reported separately. Rule catalog: R-STMT-SET, -ABEND, -NAME, -ADDR, -BAL, -FICO, -TXN, -TOTAL, -OVERFLOW; R-REPT-SET, -AMT, -XREF, -LOOKUP, -ACCT, -LASTACCT, -LASTDUP, -GRAND, -HEADER; R-ABSENT. Statements a build prints for cards legacy never reaches on `c03` are R-STMT-ABEND. Platform assumptions: S1 the output pad bytes are never read; S2 equal sort keys keep transaction-ID order, so page totals and page breaks are a layout score, never equivalence; S3 the report window; S4 the overflow statement excluded; S5 the `c03` statements before the abend kept. Validation before any layer 1 build: 18 planted defects, all detected as expected (`scorer/report_selftest.py`); golden against itself 100% strict and business on all five sets; the INTCALC self-test still passes [notebook §44]. Flags are mapped as above, from each build's register, before its outputs are opened (`runs/layer1/builder_flags.json`, hashes in `runs/FLAGS_LAYER1.sha256`). As for POSTTRAN, a register entry about input format is not mapped to a rule.

3.6 Provenance of the checks

The authors had seen both wikis before any check existed, so no check is blind to them. PROTOCOL §3a records, for each check, whether it came from the COBOL before the first close read of the interest pages or after. The last-account defect and the fallback were found from the COBOL first; the rule catalog and hypotheses H1–H4 were written after the read. The primary endpoint does not depend on any named check: it covers every record and every business field. H1–H4 are therefore *informed* predictions, not blind ones.

3.7 An online oracle (v1 used by A24; v2 prepared, not scored)

The pre-registered design covers batch steps only; the online (CICS) half of CardDemo was out of scope because it could not run (PROTOCOL §10). We have since built an online oracle for a first slice of four screens: sign-on (COSGN00C), the main menu (COMEN01C), account view (COACTVWC) and bill payment (COBIL00C) [`golden-online/v1/RESULTS.md`].

- **How it runs.** The four programs run unmodified under GnuCOBOL 3.2 with the batch oracle's flags. Each `EXEC CICS` is turned into a call to a CICS/BMS stub we wrote in COBOL, which models the file commands the slice uses (reads, alternate-index paths, browses, duplicate keys), the BMS screen model and the 3270 input rules, and a fixed clock. A session driver runs each screen interaction as a new task and cancels each program after it runs, as pseudo-conversational CICS starts each task with fresh working storage. Each rule cites the vendor's reference entry it comes from.

- **Instrument checks.** 48 stub self-tests pass (files, browses, empty files, BMS input and output, clock and program control, and one data-conversion check). 17 sessions (3 sign-on, 2 menu, 5 account view, 7 bill payment), each with its targeted branches and expected outcome written down, were committed before the first oracle run and frozen as `golden-online/v1` with hashes. Against those predictions 14 sessions match and 3 match partly: two of our predictions did not hold, and one session exposed a 3270 rule the stub did not yet model, which was added with a self-test before the golden outputs were frozen. Statement coverage over the four programs is 78.6% (550 of 700 statements); what is not covered needs fault injection, is unreachable in the slice, or is avoided on purpose because the legacy code reads outside a table [`golden-online/v1/RESULTS.md`, `golden-online/v1/COVERAGE.md`].
- **End to end.** A bill payment made on the oracle's screens (account 00000000001, balance 194.00) was carried through the batch chain: the online transaction joins through the backup that COMBTRAN merges, as on the mainframe. Every changed line of the batch outputs is explained: the account balance falls by 194.00 through POSTTRAN and INTCALC, the payment appears on the account's statement and in the daily report, and the totals move by 194.00. Interest is unchanged, because it is computed from the category balances. One change is a legacy property, not an artefact of the oracle: the payment is stored as a positive amount, so the statement total adds it. With no online session, the same chain reproduces every `golden/v1` output byte for byte.
- **Limits.** There is no reference CICS: every CICS and BMS rule is our reading of the vendor's documentation, and the self-tests show that the stub does what we wrote, not that CICS does it. Two rules are explicit assumptions with no vendor text (an empty-file browse and a key change before a backward read); one session rests on the first and is kept out of scoring, and no session uses the second. Two places read uninitialised working storage, which GnuCOBOL fills with X'00' and z/OS leaves undefined (a card number in one bill-payment error path and a filler area); both are excluded from scoring. Only the commands the slice uses are modelled; anything else stops the run.

Used by A24. This first slice, `golden-online/v1`, is the oracle of the registered screens test (§6.2).

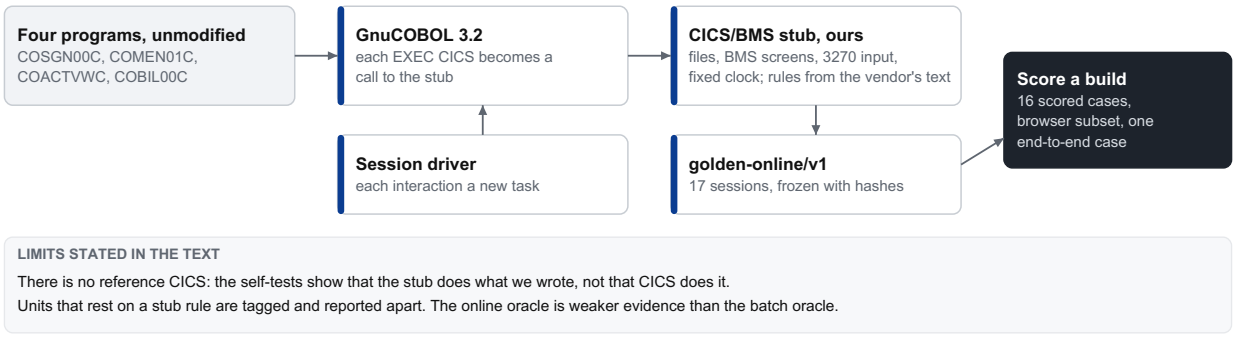

Figure 4. The online oracle, first slice (golden-online/v1), used by the registered screens test. Diagram: no data plotted.

Version 2: preparation, not scored. We then extended the oracle to all 17 base screens, adding the 13 remaining programs (card list, view and update; transaction list, view and add; account update; user list, add, update and delete; report request; admin menu) [`golden-online/v2/RESULTS.md`]. No build has been scored against it, and no hypothesis uses it.

- **Cases.** 38 online cases and 4 end-to-end cases, each with its predictions committed before its first run: 42 predictions, of which 31 match, 9 match partly and 2 do not match. Every deviation was our misreading of the source or a detail left out of a prediction; none led to a change in a stub rule. Regressions pass (38 of

38 on v2, 17 of 17 on v1), as do the self-tests (82 and 48), which show that the stub does what we wrote, not that CICS does it.

- **Coverage.** 80.3% of statements over all cases (3,624 of 4,515), and 309 of 335 paragraphs. This is statement coverage and says nothing about branches. Some paths are left out on purpose [`golden-online/v2/COVERAGE.all.md`].
- **The largest assumption: the 8-byte CEEDAYS call (rule R-CEE-PIC8).** COACTUPC calls the date service with 8-byte fields through a routine that reads 10 bytes from each, so the date picture reaches the service with two trailing blanks. The vendor's text does not settle what the service does with trailing blanks in a picture. We assume that a valid date in the first 8 bytes is accepted, the result under the two readings the text allows. A third reading is not excluded in words, and under it every COACTUPC case past a change would come out differently. The rule was written down with its source before the first COACTUPC run. It was not checked on a real mainframe, and the vendor's text was read through a page-to-text tool, not checked against a PDF.
- **Other limits.** As in v1, there is no reference CICS. Storage with no initial value is X'00' under GnuCOBOL and depends on a runtime option on z/OS; a few fields carry that difference and are recorded, not scored. Instrument events on the way to the freeze are disclosed in the results file.
- **Legacy properties recorded.** Two came up while building the cases: the "no change" path is unreachable on the sample accounts, and COACTUPC's update record has no zip field, so a save moves the group id to the zip offset. They are recorded as what the original does, not as correct behaviour. A rebuild that keeps the record intact will differ from this golden.

4 Results

4.1 Change request (A16)

[`runs/a16/SUMMARY.md`, `runs/a16/PRESCORE.md`; notebook §56, §58; PROTOCOL A16]

Design. One business rule is changed once in the Sonnet wiki and once in the COBOL. The wiki states the new rule as the rule, with no change-request marker. The edited COBOL, run under GnuCOBOL with the same flags on the five sets, gives the new golden outputs (`golden/cr1`, `golden/cr2`); independent reference models with the same one-line change agree on every set, and the negative control against `golden/v1` fails as it should.

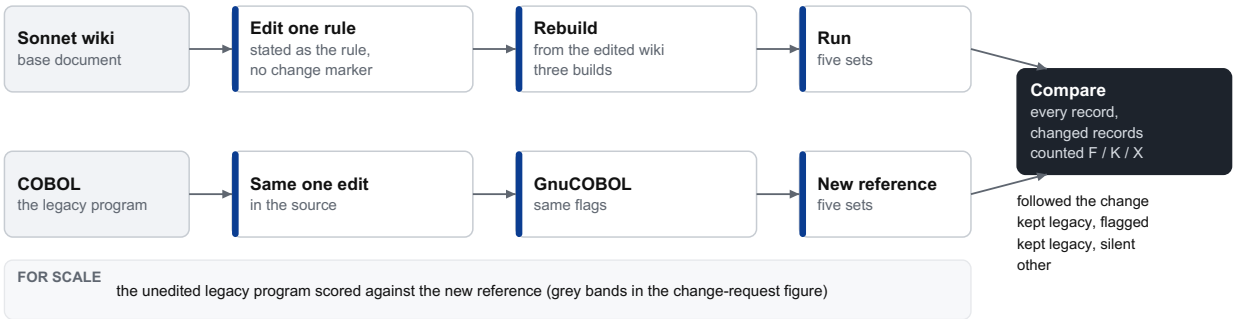

Figure 5. How a change request is run and scored: one rule is changed once in the wiki and once in the COBOL, and each rebuild is compared with the edited legacy program. F, K and X count changed records that followed the change, kept the legacy value, or differ from both. Diagram: no data plotted.

- **CR2, the headline (POSTTRAN):** a transaction is over limit only beyond the credit limit × 1.10. We know of no public source that proposes it (we kept no search log for this rule; the novelty log covers

CR3–CR8), and the public program has the old rule, so memory of CardDemo works against the build. The unedited legacy program scores 91.1–92.3% against `golden/cr2`.

- **CR1, the contrast (INTCALC):** interest is rounded to the cent (`COMPUTE … ROUNDED`) where legacy truncates. The same change is proposed in CardDemo's public tracker (pull request #32), so memory can supply it. The unedited legacy program scores 43.4–44.7% against `golden/cr1` (44.0% on base, c01 and c03, 43.4% on c02, 44.7% on c04).

Each build's outcome is one of "followed the change", "kept legacy, flagged", "kept legacy, silent" or "other". **H11** (FlowX Platform MCP) and **H11h** (Agentic Harness), tested separately and never pooled: each CR build follows the changed rule, with no silent R-INT-TRUNC (CR1) or R-POST-102 (CR2) divergence; refuted for a builder and a rule if two or more of three builds follow legacy silently. The Agentic Harness arm is reported in §4.9.

Results. Business equivalence, %; strict equals business in every cell.

Table 2. CR1 and CR2, FlowX Platform MCP: business equivalence per set, %; strict equals business in every cell.

Builder	Build	base	c01	c02	c03	c04	Divergence	Outcome
FlowX Platform MCP	CR2 r1–r3	100	100	100	100	100	none	followed the change (3 of 3)
FlowX Platform MCP	CR1 r1–r3	99.00	99.00	98.99	99.00	99.03	last account, silent (R-INT-LAST)	followed the change (3 of 3)

FlowX Platform MCP: H11 supported for both rules (0 of 3 builds follow legacy silently, for CR1 and for CR2). The A16 expectation holds as written: CR1 about 99% with R-INT-LAST the only silent rule, CR2 100%. All three CR2 builds implement `creditLimit × 1.10` with a strict > in integer cents; r1 adds that from memory CBTRN02C has no tolerance and r2 calls the rule the one most likely to differ from legacy, both flagging R-POST-102, while r3 raises no uncertainty. All three CR1 builds round to the cent, halves away from zero, and also flag that legacy may truncate; r1 cites its recollection of CBACT04C as "unverified". No build changed a rule back to what it recalled. The one CR1 divergence per set is the legacy last-account record, the same silent divergence as in the undisguised INTCALC Sonnet-wiki builds; no CR1 register raised it. Flags raised per build, none needed: CR1 14 / 9 / 9, CR2 12 / 14 / 11.

Disclosures. The bundle file name (`cr1-…`, `cr2-…`) is visible to the builder; two CR2 registers name the file as the source of the wiki, and no register mentions a change request. One CR2 build's flags (FlowX Platform MCP r2) were revised before its execution, with no output opened, so that one statement maps the same way as in r3; both hashes are kept and the later file is scored [`runs/a16/PRESCORE.md`, `runs/a16/FLAGS_A16.sha256`].

Six further change requests (A21). [PROTOCOL A21, A22; notebook §62–§64; `runs/a21/`] Two rules on one builder are thin, so six more rules were prepared and pre-registered before any of their builds, one or two on each step, prepared as CR1 and CR2 were. None was found proposed in CardDemo's public tracker or on the web, in 44 searches; nothing stronger is claimed. This time the bundle files and project names carry no change-request marker.

Table 3. The six further change requests (A21), with the unedited legacy program against each new golden.

CR	Step	The edited rule	Unedited legacy vs the new golden, business %
CR3	INTCALC	Accounts opened ten or more years before the run date pay the group rate minus 2.00, never below 0	70.7–71.8
CR4	POSTTRAN	At most four purchases per card per run; a fifth that passes the other checks is rejected 104	72.0–72.7
CR5	POSTTRAN	Transactions post in ascending order of amount, credits first; equal amounts keep feed order	84.4–84.7
CR6	CREAST-MT	The statement total counts charges only; credits are listed but not subtracted	50.0
CR7	TRANREPT	Interest postings are left out of the daily report and its totals	72.4–73.2
CR8	CREAST-MT	The statement shows the FICO band (Poor to Exceptional) in place of the score	0.0

Each rule's oracle comes from the patched COBOL; an independent model written from the rule text agrees with it on all five sets, and fails against `golden/v1`. **H14** (FlowX Platform MCP) and **H14h** (Agentic Harness, §4.9), tested separately: per rule, refuted if two or more of three builds follow the legacy rule silently; overall, refuted for a builder if that happens on two or more rules. Under the registered flag rubric a wiki TODO counts as a flag for every build, and the base wikis carry TODOs on the CR6 and CR7 rules, so no CR6 or CR7 build can be "kept legacy, silent": H14 is informative on CR3, CR4, CR5 and CR8. A build-flags-only outcome for CR6 and CR7, defined before their builds, is reported as supplementary. Known limits stated before any build: the CR8 field is still labelled "FICO score" in the file contract; CR5 interacts with the legacy over-limit sign; CR3 depends on the run date.

Results, FlowX Platform MCP [notebook §67]. Business equivalence, %, per build over base / c01 / c02 / c03 / c04.

Table 4. Six further change requests, FlowX Platform MCP: business equivalence per build.

CR	r1	r2	r3	Outcome	What differs
CR3	99.00 / 99.00 / 98.99 / 99.00 / 99.03	same	same	followed ×3	the legacy last account (R-INT-LAST): flagged in r1, silent in r2 and r3
CR4	100 on every set	100	100	followed ×3	none
CR5	100 on every set	100	100	followed ×3	none
CR6	0 on every set	0	100 / 100 / 100 / 88 / 100	followed ×1, other ×2	r1, r2: address line 3 without the country code, flagged, on every statement; r3: the `c03` card on which legacy abends, flagged
CR7	99.68 on every set	99.68	99.68	followed ×3	the legacy end-of-file re-add of the last amount in the grand total, not copied, flagged
CR8	100 / 100 / 100 / 88 / 100	same	same	followed ×3	the `c03` abend card, flagged

FlowX Platform MCP: H14 not refuted (0 of 6 rules; no build kept the legacy rule, flagged or silent). 16 of 18 builds followed the change; the build-flags-only outcome is the same on all 18. CR6 r1 and r2 are "other" under the registered record-level rule: their total field equals `golden/cr6` on all 243 changed records, but each of those records also carries the unrelated address difference. The rule cannot separate the two, a limit found when CR6 r1 was scored, after A21 was stamped. A supplementary per-field check, defined after that

point and never used for H14, finds the changed field equal to `golden/crN` on every changed record of all 18 builds. The 18 builds cost $57.00 in all (mean $3.17) and 6.7 minutes each on average.

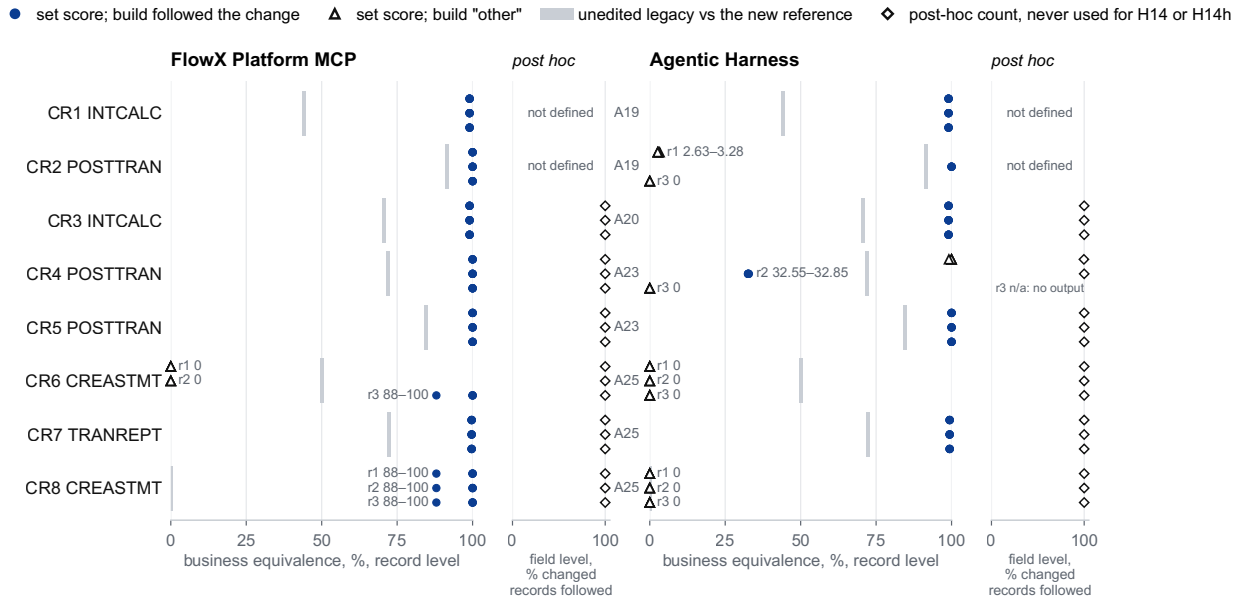

Figure 6. Change-request outcomes, CR1–CR8, each build shown and never a mean. Each dot is one data set's business equivalence, as registered at record level; rows hold r1, r2, r3 top to bottom. n = 48 builds, 24 per builder, five sets each. Grey bands give the unedited legacy program against the new reference. CR6 and CR8 stay at 0% in all six Agentic Harness builds, and CR6 r1 and r2 in the FlowX Platform MCP arm: every statement differs from legacy on a flagged address line. Post hoc: the narrow columns show the field-level count, defined after A21 was stamped and never used for H14 or H14h; no such count was defined for CR1 and CR2. Labels beside the Agentic Harness panel name the amendment whose version each rule ran on. Strict differs from business only on Agentic Harness CR7 (by at most 0.63 points) among these builds.

4.2 Wiki ablation under disguise (A17)

[`runs/a17/SUMMARY.md`, `runs/a17/PRESCORE.md`; notebook §57; PROTOCOL A17]

Design. On the disguised INTCALC (A5: every identifier renamed, five values perturbed so that recall gives the legacy value), four arms differ only in the specification: *detailed* (the A5 Sonnet-wiki bundle), *summary* (the process page's property table, purpose and process flow, stopping before its Business Rules section, plus the record layouts and contract), *no wiki* (record layouts and contract), and *code* (the disguised COBOL with its copybooks and shims, plus the contract). Three builds per arm, interleaved per rep, scored against the disguised oracle. The no-wiki arm is meaningful only under disguise; on public CardDemo it would measure memory. **H12:** the detailed arm's median business equivalence is no more than 1 point below the code arm's. **H13:** the detailed arm is more than 1 point above the summary arm. **Instrument check:** the no-wiki arm stays more than 5 points below the detailed arm, else the ablation is uninformative. The primary figure is the median over three builds of each build's mean over the five sets.



Figure 7. The ablation under disguise (A17): on the disguised INTCALC, four arms differ only in the specification. Diagram: no data plotted.

Results, FlowX Platform MCP. Business equivalence, %; strict in brackets where it differs.

Table 5. Wiki ablation under disguise, FlowX Platform MCP: business equivalence per arm, strict in brackets where it differs.

Arm	Builds (base)	Median of build means	base	c01	c02	c03	c04	Silent divergences
detailed	99.00 / 99.00 / 99.00	99.004	99.00	99.00	98.99	99.00	99.03	last account in r2 (r1, r3 flagged)
sum-mary	99.00 (49.00) / 99.00 (49.00) / 2.00 (1.00)	99.004	99.00	99.00	98.99	99.00	99.03	none
no wiki	1.00 / 2.00 / 2.00	2.396	2.00	2.00	4.04	2.00	1.94	none
code	100 / 100 / 100	100.000	100	100	100	100	100	none (legacy behavior reproduced from the code)

- **H12 supported, at the margin.** Code minus detailed is 0.996 points on the pre-registered figure, inside the 1-point limit. Read per set the gap is 1.00 / 1.00 / 1.01 / 1.00 / 0.97, so on c02 alone it is 0.01 over the limit. Both readings are reported. The whole gap is the last-account record: the code arm reproduces the legacy defect from the code without comment, while the detailed arm follows the wiki, which has every account updated, and two of its three builds flag that legacy may differ.
- **H13 not supported.** Detailed minus summary is 0.00 on every set. Summary r1 and r2 equal detailed in business terms but fill the transaction source and description, which their bundle omits, from recall of CardDemo, so their strict score is about 49%. Summary r3 divides by 12,000 where the formula needs 120,000 and posts ten times the legacy interest, with every differing record flagged.
- **Instrument check passes:** detailed minus no wiki is 96.6 points on the pre-registered figure, and more than 5 on every set. Caveat: all three no-wiki builds identified the program as CardDemo and filled the gaps from recall (the `DEFAULT` fallback and the legacy constants), which the disguise makes differ from the oracle. The gap therefore measures the wiki against recall, not against a builder with no prior.
- The A5 detailed results (99 / 99 / 100 on base) are reported beside this rerun, not pooled with it.

The Agentic Harness arm of the ablation is reported in §4.9.

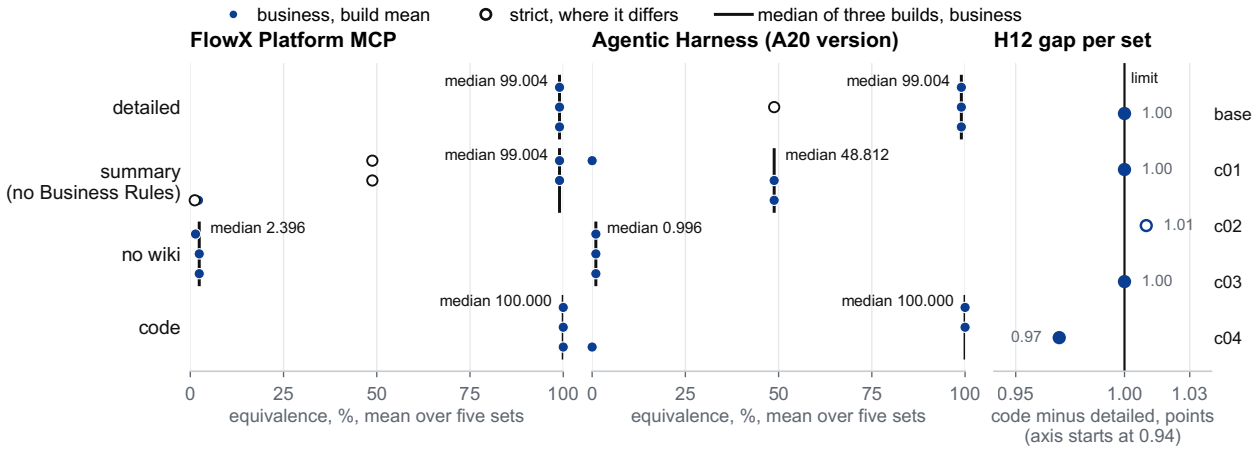

Figure 8. Wiki ablation under disguise (A17): the score with and without the Business Rules section. Each dot is a build's mean over five sets, the tick the pre-registered median of three builds, on business equivalence; n = 12 builds per builder. The Agentic Harness arm is on the A20 version. Open dots give the strict score where it differs. Without the section two of three FlowX Platform MCP builds filled the gap from recall of CardDemo, and the Agentic Harness builds lost 50 points. The no-wiki builds lean on recall too, so that gap measures the wiki against recall. Right: H12's gap per set, the same for both builders; the open dot is c02, 0.01 over the limit. Not post hoc.

4.3 FlowX Platform MCP builds (secondary condition)

[notebook §25–26]

Table 6. INTCALC, FlowX Platform MCP (secondary condition): business equivalence per set, %.

Build	base	c01	c02	c03	c04	Divergence	Flagged?
FlowX Platform MCP × Sonnet r1	99.00	99.00	98.99	99.00	99.03	last account	silent
FlowX Platform MCP × Sonnet r2	99.00	99.00	98.99	99.00	99.03	last account	flagged
FlowX Platform MCP × Sonnet r3	99.00	99.00	98.99	99.00	99.03	last account	flagged
FlowX Platform MCP × Opus r1	100	100	100	100	100	none	n/a
FlowX Platform MCP × Opus r2	99.00	99.00	98.99	99.00	99.03	last account	flagged
FlowX Platform MCP × Opus r3	99.00	99.00	98.99	99.00	99.03	last account	flagged

Business equivalence, %. In each divergent build the one record that differs on every set is account 00000000050: the build adds the interest (+18.77) and resets the cycle totals, as both wikis describe; legacy does neither. Every interest amount matches, including truncation, the zero-rate skip, negative interest and the fallback group.

Five of six builds followed the documented behavior on the last account; four of those five warned that legacy might differ there. Sonnet r2 and r3 raised it by reasoning about the wiki's wording ("at each change of account"); Opus r2 and r3 raised it from recall. Opus r1 reproduced the legacy defect on purpose, from recall.

The recall confound. All three Opus-wiki builds state that they filled the wiki's gaps (the formula, the fallback) from memory of the public CardDemo source, which was not in their bundle [notebook §25]. File isolation cannot block recall of training data. So these scores measure wiki plus recall.

4.4 Recall control: the disguise (A5)

To separate the two, amendment A5 (written after the scores above, before any disguised build) renamed every CardDemo identifier and perturbed five values where recall would supply the legacy value: interest type and category, source, description prefix, and the fallback group (`DEFAULT` → `STANDARD`, in the program and in the rate data). The arithmetic, layouts and control flow, including the last-account defect, are unchanged. The disguised COBOL gives a second oracle, `golden/d1`, which differs from `golden/v1` in exactly the four per-

turbed transaction fields on all 252 interest transactions and nowhere else [notebook §28]. The Sonnet bundle states all perturbed values; the Opus bundle states none.

[runs/A5_SUMMARY.md]

Table 7. The A5 disguise: FlowX Platform MCP builds against the disguised oracle.

Build (disguised)	Overall	Legacy values in outputs	Entries citing CardDemo
FlowX Platform MCP × Sonnet r1 / r2 / r3	99.0 / 99.0 / 100	0 / 0 / 0	0 / 0 / 0
FlowX Platform MCP × Opus r1 / r2 / r3	1.4 / 1.0 / 1.4	4 / 4 / 4	4 / 9 / 2

With every value stated, the Sonnet-wiki builds behave as before and show no recall (H6 supported). With the values left out, all three Opus-wiki builds identified the disguised code as CardDemo, fell back to the recalled DEFAULT group, which does not exist in the disguised data, and priced almost nothing. Their one priced transaction has the legacy amount and four recalled field values (H5 not supported; H7 supported, with recall leaking through the disguise). Sonnet r3 matched legacy exactly by reading the wiki's "at each change of account" literally as a control break: the legacy defect reproduced from the wiki's own wording, without citing CardDemo.

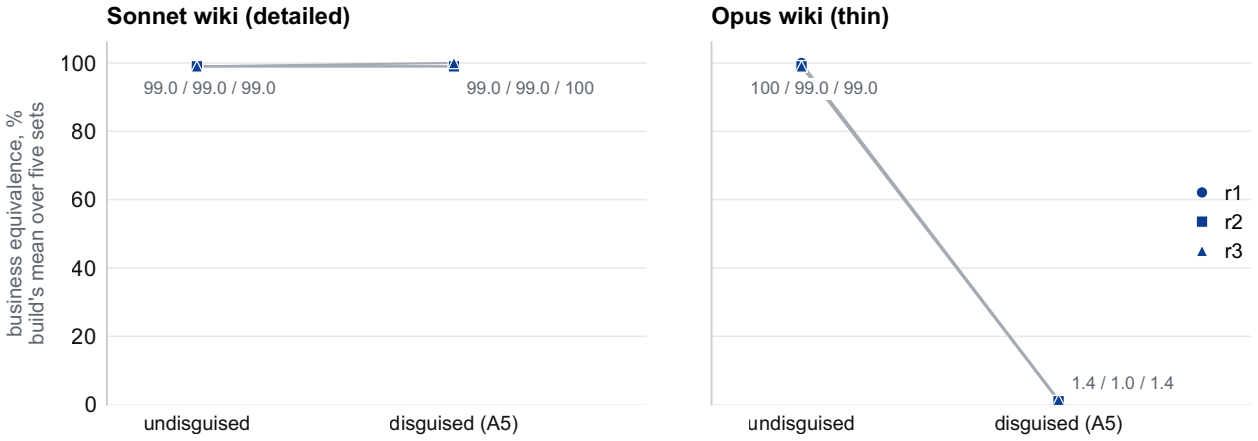

Figure 9. Recall control: FlowX Platform MCP on INTCALC, undisguised and under the A5 disguise, three builds per wiki (n = 12). Value labels list r1 / r2 / r3; markers tell the builds apart. A5 was written after the undisguised scores and before any disguised build. Not post hoc.

4.5 Ontology arm (A6, A6b)

A FlowX.AI ontology built from the disguised source was added to the bundle (A6). With it, two of three Opus-wiki builds reached 99.0%: the fallback group's literal is readable from a procedure name the ontology carries, and the transaction constants come from the ontology [runs/A6_SUMMARY.md]. **This ontology was re-extracted after the first extraction had been inspected** and judged an under-extraction [notebook §31]. An exploratory arm (A6b) built against the withdrawn first extraction: Opus-wiki builds returned to about 1%, Sonnet-wiki builds were unchanged [runs/A6B_SUMMARY.md]. The A6 Opus gain therefore comes from content only the re-extraction holds, and must be read with that choice beside it. The number of data-meaning assumptions fell with either ontology (H8, descriptive), so that drop does not track the ontology's rule content.

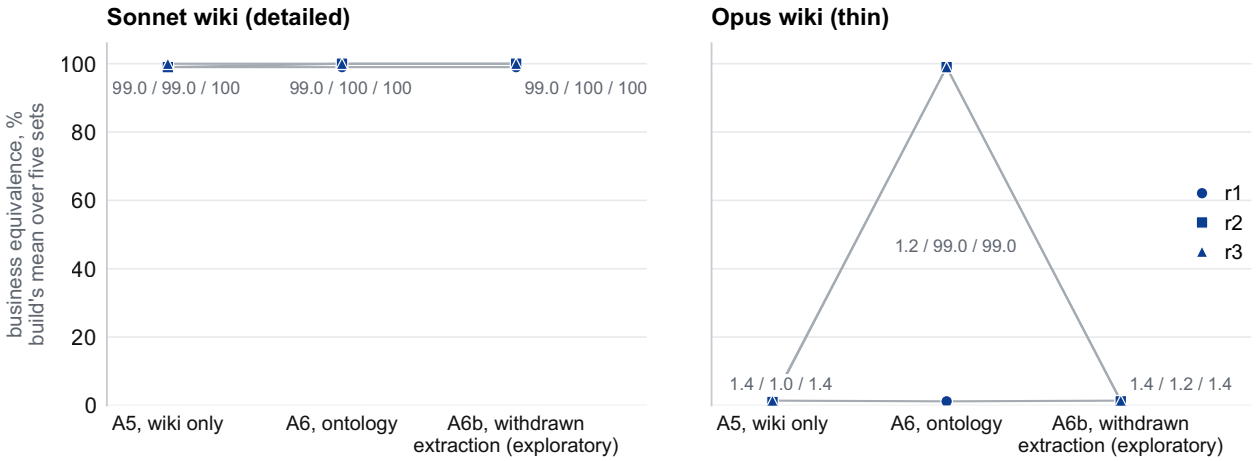

Figure 10. Ontology arm on the disguised INTCALC, FlowX Platform MCP, three builds per wiki per arm (n = 18). The A6 ontology was re-extracted after the first extraction had been inspected; A6b, on the withdrawn first extraction, is exploratory. Value labels list r1 / r2 / r3.

4.6 Hypotheses

H1–H4 were pre-registered against the primary condition and its Opus counterpart (PROTOCOL §8). The primary condition produced no records, so its rule tables hold only R-ABSENT. We report each hypothesis there first, then on the FlowX Platform MCP builds, which the hypotheses also cover ("every build").

Table 8. Hypotheses H1–H4 on the primary condition and on FlowX Platform MCP.

	Prediction	Agentic Harness (primary)	FlowX Platform MCP
H1	every build diverges on the last account	not supported: no build produced the record (R-ABSENT)	not supported: 5 of 6 (Opus r1 reproduces the defect from recall)
H2	that divergence is silent in every build	not supported: no R-INT-LAST divergence exists to classify	not supported: silent in 1 of 5
H3	Sonnet-wiki builds truncate, or diverge on truncation flagged	not assessable: no amounts produced	supported: no truncation divergence
H4	Opus-wiki builds diverge on the rate formula	not supported: no amounts produced	not supported: the formula was recalled (§4.4)

H5–H10 (A5, A6) are reported in §4.3–4.4 and in the arm summaries; H11 (change request) in §4.1; H12 and H13 (ablation) in §4.2; the Agentic Harness hypotheses H11h–H19h in §4.9.

4.7 Second step: POSTTRAN (A11, secondary)

[`runs/POSTTRAN_SUMMARY.md`; notebook §40–43]

A11 was written after every INTCALC build was scored and before any POSTTRAN build: same builders, wikis, prompt (the step name is its only slot), gate rules and scoring; FlowX Platform MCP first, then Agentic Harness (§4.9); no disguise or ontology arm; no new hypothesis.

Table 9. POSTTRAN (A11, secondary), FlowX Platform MCP: business equivalence per set, %; strict is identical in every cell.

Build	base	c01	c02	c03	c04	Divergent records, base (silent / flagged)	Divergent rules
FlowX Platform MCP × Sonnet r1–r3	100	100	100	100	100	0	none
FlowX Platform MCP × Opus r1	100	100	100	100	100	0	none
FlowX Platform MCP × Opus r2	87.18	84.73	87.18	87.21	87.37	0 / 60	102, 103, BAL, PRECEDENCE, TCAT
FlowX Platform MCP × Opus r3	87.18	85.74	87.18	87.21	87.37	0 / 60	102, BAL, TCAT

Business equivalence, %; strict is identical in every cell.

Sonnet wiki. Every FlowX Platform MCP build matches legacy on every record of every set.

Opus wiki. Builds that follow the wiki's credit rule post about 12 more transactions per set than legacy, with the balance and category-balance differences that follow, and flagged every divergent rule beforehand. FlowX Platform MCP Opus r2 and r3 took the wiki's rule as their default and kept the recalled legacy rule as an option; FlowX Platform MCP Opus r1 chose the recalled legacy rules and scores 100%. All three cite CBTRN02C from memory (11, 8 and 7 mentions), so, as in §4.3, the FlowX Platform MCP Opus-wiki result cannot be separated from recall.

Honesty. In the FlowX Platform MCP POSTTRAN builds, every divergence is flagged.

4.8 Layer 1: CREASTMT and TRANREPT (A12–A14, secondary)

[`runs/LAYER1_SUMMARY.md`; `runs/layer1/*_scores.md`; notebook §44–50]

A12–A14 keep the builders, wikis, prompt (the step name is its only slot), gate rules and flag procedure; FlowX Platform MCP first, then Agentic Harness (§4.9); no disguise or ontology arm, no repair, no new hypothesis. Strict and business equivalence are given separately, as `strict / business`.

CREASTMT (statements, text and HTML; equivalence in %; `c04` with the overflow statement excluded, S4)

Table 10. CREASTMT (A13), FlowX Platform MCP: equivalence per set, strict / business, %; c04 with the overflow statement excluded (S4).

Build	base	c01	c02	c03	c04	Divergence	S4 overflow units differing
FlowX Platform MCP × Sonnet r1	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	none	2 of 2
FlowX Platform MCP × Sonnet r2	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	none	2 of 2
FlowX Platform MCP × Sonnet r3	100 / 100	100 / 100	100 / 100	88 / 88	100 / 100	c03: 12 extra units, flagged (R-STMT-ABEND)	2 of 2
FlowX Platform MCP × Opus r1	100 / 100	100 / 100	100 / 100	88 / 88	100 / 100	same, flagged	2 of 2
FlowX Platform MCP × Opus r2	100 / 100	100 / 100	100 / 100	88 / 88	100 / 100	same, flagged	2 of 2
FlowX Platform MCP × Opus r3	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	none	0 of 2

Units per set: 100 on base, c01, c02; on c03, 88 where the build stops with legacy and 100 where it prints the extra statements; 98 on c04. Flags raised per FlowX Platform MCP build: 20–26, none needed on base.

TRANREPT (daily transaction report; equivalence in %; 362–370 units per set)

Table 11. TRANREPT (A14), FlowX Platform MCP: equivalence per set, strict / business, %; 362–370 units per set.

Build	base	c01	c02	c03	c04	Divergence
FlowX Platform MCP × Sonnet r1–r3	99.72 / 99.72	99.73 / 99.73	99.72 / 99.72	99.72 / 99.72	99.73 / 99.73	grand total, flagged (R-REPT-LASTDUP)
FlowX Platform MCP × Opus r1–r3	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100	none

The three FlowX Platform MCP builds per wiki give identical scores on every set. Flags raised per FlowX Platform MCP build: 22–23 (Sonnet), 16–19 (Opus).

FlowX Platform MCP. All twelve builds return output on every set. On CREASTMT every build matches legacy on every statement of four sets, the city truncation included. On `c03` the builds split on a rule the wikis leave open: Sonnet r1, Sonnet r2 and Opus r3 stop where legacy abends; Sonnet r3, Opus r1 and Opus r2 skip the orphan card and print six statements legacy never writes (12 units), each having recorded that choice. The excluded overflow statement differs in five builds; Opus r3 reproduces it exactly, by emulating CBSTM03A's 51 × 10 table from what it calls its recollection of the source [notebook §46]. On TRANREPT the builds split by wiki. The Opus-wiki builds reproduce CBTRN03C's end-of-file behavior from recall and say so: 100% on every set. The Sonnet-wiki builds name the same legacy defect and decline to copy it, following the wiki's totals; the grand total is their one divergent unit per set, flagged. Every detail line and every account total matches in all six. All six take legacy's 20-line page from recall, since neither wiki states it (page totals are layout only, S2) [notebook §49].

Strict and business. They are equal in every FlowX Platform MCP build. The business parser does not count label spacing or the column-header line, so layout differences count against strict and the layout line match only (§7, PDF whitespace collapse).

Honesty. Across the 12 FlowX Platform MCP layer 1 builds, no build differed from legacy without a flag.

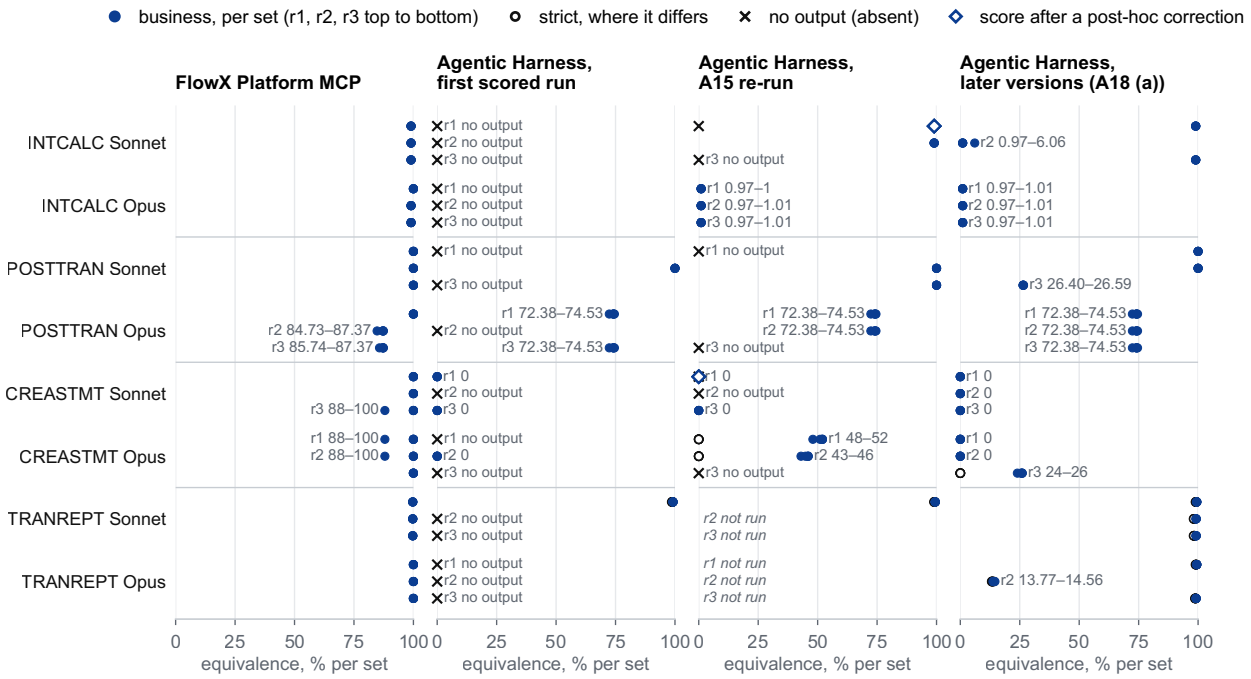

Figure 11. Rebuilds with no rule changed: baseline equivalence per program, wiki and builder. Each filled dot is one set's business equivalence, each open dot its strict score where the two differ by more than half a point; r1, r2, r3 top to bottom. n = 24 builds per column, except the A15 re-run: 19 run and 5 TRANREPT builds not run, by decision. The A15 re-run and the later versions (A18 (a), 4 builds on the A30 version and 20 on the A31 version) are exploratory and test no hypothesis; the columns are never pooled. Diamonds mark the two post-hoc decoding corrections, INTCALC Sonnet r1 and CREASTMT Sonnet r1; each is absent as first decoded (the cross). Matches on the Opus wiki lean on recall of the public source.

4.9 The second registered builder (Agentic Harness)

[runs/POSTTRAN_SUMMARY.md, runs/LAYER1_SUMMARY.md, runs/RERUN_SUMMARY.md, runs/a19h/SUMMARY.md, runs/a20h/a17/SUMMARY.md, runs/a21/Agentic Harness/SUMMARY.md, runs/a23/Agentic Harness/SUMMARY.md, runs/a25/Agentic Harness/SUMMARY.md, runs/a30/a18a/, runs/a31/a18a/, runs/a30/a24h/r1/SCORE.md, runs/a31/a24h/r2/SCORE.md; PROTOCOL A15–A31]

The protocol registered a second builder from the start: the Agentic Harness (§3.3). The originally registered primary condition is one of its arms. Each arm ran on a builder version named in an amendment timestamped before its builds, and results are reported per version: never pooled with FlowX Platform MCP, and never pooled across versions except the H14h rule count, as registered. The two builders used different models. The second builder's full execution record is in the repository (results/HARNESS_STOPS.md).

Table 12. The second registered builder: every registered arm, on the version named in each row. Never pooled with FlowX Platform MCP.

Arm	Version	Outcome
Four steps, first scored run; primary condition (INTCALC, Sonnet wiki), H1–H4	first study version	7 of 24 builds returned output; the primary condition returned no records, so H1, H2 and H4 are not supported and H3 is not assessable there (§4.6)
Four steps, re-run (A15, exploratory)	A15	19 of 24 run (5 not run, by decision); 12 returned output as first decoded, 14 with two labelled post-hoc decoding corrections (§7.1)
CR1, CR2 (A16)	A19	CR1 followed in 3 of 3 (98.99–99.03%); CR2 put in code in 3 of 3, every record matched in 1 of 3. H11h supported for CR1; for CR2 not refuted, the per-build criterion met by 1 of 3
Wiki ablation under disguise (A17)	A20	medians: detailed 99.004, summary 48.812, no wiki 0.996, code 100.000. H12 supported at the margin; H13 supported
Six further rules (A21)	A20, A23, A25	18 counted builds: 10 followed the change, 8 "other", none kept legacy silently; H14h not refuted on each rule and overall
Four steps on the latest versions (A18 (a), exploratory)	A30, A31	24 of 24 returned output; 7 builds have a silent divergence at base
CR6–CR8 on the latest versions (exploratory)	A28–A31	the same outcome classes as the counted builds
Four screens (A24)	A26–A31	no screen built; H15h–H18h refuted on 2 of 3 counted builds, both scored absent (with those two set aside, no verdict is reached); H19h not run
Whole base application (A32)	not registered	not part of A32

First scored run. No INTCALC build returned a record. On POSTTRAN, Sonnet r2 matches legacy on every record of every set (100%), and Opus r1 and r3 score 72.38–74.53%, every divergence flagged: two separate workflows made the same three choices, the wiki's credit rule, reason codes of their own (1–7), and cycle totals booked with the opposite sign, and their registers call codes and sign unspecified. On CREASTMT, three builds returned statements (Sonnet r1, Sonnet r3 and Opus r2); they follow the contract's line count, order, separators, field positions and pictures, but every statement differs from legacy in business fields, all flagged, address line 3 in all three. On TRANREPT, Sonnet r1 scores 98.65–98.91% strict and 99.17–99.19% business: every detail line and every account total legacy prints matches in business, and its three divergent units per set (the grand total, an account total for the last card that legacy omits, and the header's date range) are flagged. On POSTTRAN, CREASTMT and TRANREPT no build that returned output differed from legacy without a flag. A build or set that returned nothing is scored R-ABSENT, silent, as the protocol requires.

Re-run (A15, exploratory). Every build was rebuilt once under A15, with the same wikis, bundles, prompt, gate rules, adapters, scorers and flag procedure; the TRANREPT re-run was stopped by decision after its first build. The two runs are never pooled. Business equivalence, % per build over the five sets (CREASTMT and TRANREPT as strict / business).

Table 13. Agentic Harness: first scored run and A15 re-run, per build; never pooled.

Step	Build	First scored run	Re-run (A15)
INTCALC	Sonnet r1	0, no output	0 as first decoded; 98.99–99.03 after a post-hoc decoder correction (§7.1); last account silent
	Sonnet r2	0, no output	98.99–99.03; last account silent
	Sonnet r3	0, no output	0, no output
	Opus r1–r3	0, no output	0.97–1.01: every account returned, no interest posted; balance unchanged silent, the rest flagged
POSTTRAN	Sonnet r1	0, no output	0, no output
	Sonnet r2	100	100
	Sonnet r3	0, no output	100
	Opus r1	72.38–74.53, all flagged	72.38–74.53, all flagged
	Opus r2	0, no output	72.38–74.53, all flagged
	Opus r3	72.38–74.53, all flagged	0, empty lists returned, all flagged
CREAST-MT	Sonnet r1	0 / 0, all flagged; c03 absent	absent as first decoded; 0 / 0, all flagged, with a post-hoc keymap (§7.1)
	Sonnet r2	absent	absent
	Sonnet r3	0 / 0, all flagged; c03 absent	0 / 0, all flagged
	Opus r1	absent	0 / 48–52, all flagged
	Opus r2	0 / 0, all flagged; c03 absent	0 / 43–46, all flagged
	Opus r3	absent	absent
TRANREPT	Sonnet r1	98.65–98.91 / 99.17–99.19, 3 units flagged	98.92–99.18 / 99.45–99.46, 2 units flagged
	Sonnet r2–r3, Opus r1–r3	absent	not run (stopped by decision)

On the Sonnet wiki, INTCALC r2 (and r1 after the correction) score 98.99–99.03 and POSTTRAN r2 and r3 score 100, the FlowX Platform MCP scores on those steps. Among builds that returned output, silent divergences occur only on INTCALC: the last account in the two Sonnet builds with output, and the unchanged balances in the three Opus builds.

Latest versions (A18 (a), exploratory). Same wikis, bundles, prompt, gate rules, adapters, scorers and flag procedure; Sonnet r1 of each step ran on the A30 version, the other 20 builds on the A31 version; flags were mapped from each build's own register and committed before execution. Business equivalence, % per build over the five sets (CREASTMT and TRANREPT as strict / business).

Table 14. Agentic Harness on the A30 and A31 versions (A18 (a), exploratory), per build.

Step	Sonnet wiki	Opus wiki
INTCALC	r1 98.99–99.03 (last account silent); r2 0.97–6.06 (48 silent records at base); r3 98.99–99.03 (flagged)	r1–r3 0.97–1.01 (flagged)
POSTTRAN	r1, r2 100; r3 26.40–26.59 (flagged)	r1–r3 72.38–74.53 (6 silent records at base each)
CREAST-MT	r1–r3 0 / 0 (flagged)	r1, r2 0 / 0 (flagged); r3 0 / 24–26 (flagged)
TRANREPT	r1 98.92–99.18 / 99.45–99.46; r2, r3 98.11–98.37 / 99.17–99.19 (flagged)	r1 98.92–99.18 / 99.45–99.46 (1 silent unit); r2 13.22–14.02 / 13.77–14.56 (flagged); r3 98.65–98.91 / 99.17–99.19 (1 silent unit)

All 24 builds returned output on every set. 7 builds have a silent divergence at base: INTCALC Sonnet r1 and r2, POSTTRAN Opus r1–r3 and TRANREPT Opus r1 and r3. Every CREASTMT build differs on the third address line or the name, all flagged. The three runs are never pooled, and none replaces another.

Change request, CR1 and CR2 (A16, on the A19 version). H11h is tested on these builds only.

Table 15. Agentic Harness, CR1 and CR2 (A16, on the A19 version): business equivalence per set, %.

Build	base	c01	c02	c03	c04	Divergence	Outcome
CR2 r2	100	100	100	100	100	none	followed the change
CR2 r1	2.67	3.28	2.67	2.88	2.63	postings missing	other: rule changed in code
CR2 r3	0	0	0	0	0	empty output	other: rule changed in code
CR1 r1–r3	99.00	99.00	98.99	99.00	99.03	last account (R-INT-LAST): r1, r2 silent; r3 flagged	followed the change (3 of 3)

- **CR1:** all three builds round each category to the cent, halves away from zero, and raise no doubt about truncation; no R-INT-TRUNC record differs from `golden/cr1`. The three builds equal the FlowX Platform MCP CR1 builds on every set.
- **CR2:** all three workflow definitions hold the edited rule (`credit − debit + amount > 1.10 × limit`, integer cents, strict >), and none flags the tolerance. No build follows the legacy rule. r2 matches `golden/cr2` on every record; r1 and r3 post nothing on the test inputs, so their silent R-POST-102 records are `golden/cr2` rejects these builds never reach, not the legacy rule. Read strictly ("no silent R-POST-102 divergence"), only r2 meets the criterion.
- No Agentic Harness register mentions a change request.

Six further change requests (A21, on the A20, A23 and A25 versions). Which builds count was fixed before them: CR3 on the A20 version, CR4 and CR5 on the A23 version, CR6–CR8 on the A25 version.

Table 16. H14h: outcome counts per rule. The overall count spans three versions, as registered.

CR	Version	Builds scored	kept legacy, silent	kept legacy, flagged	followed	other	H14h on this rule
CR3	A20	3 of 3	0	0	3	0	not refuted
CR4	A23	3 of 3	0	0	1	2	not refuted
CR5	A23	3 of 3	0	0	3	0	not refuted
CR6	A25	3 of 3	0	0	0	3	not refuted
CR7	A25	3 of 3	0	0	3	0	not refuted
CR8	A25	3 of 3	0	0	0	3	not refuted

Overall H14h: not refuted (0 of 6 rules). Across the 18 counted builds, none kept legacy silently on its rule's changed records; 10 followed the change and 8 are "other". CR3 (99.00 / 99.00 / 98.99 / 99.00 / 99.03, the legacy last account silent in each, as FlowX Platform MCP), CR5 (100 on every set) and CR7 (business 99.36–99.37%) followed the change in 3 of 3 builds. On CR4, r2 followed the change, r1 is "other" (its purchase test also requires transaction type 01, which the rule does not; the one place it kept a legacy result, on c04, was flagged in advance), and r3 returned no output. CR6 and CR8 are "other" in 6 of 6 builds and 0% at record level: every statement differs from legacy on a flagged address line, which makes every changed record differ from both legacy and the new golden. Only the post-hoc field-level count (defined after A21 was stamped, never used for H14h) shows the changed rule followed there: CR4 r1 669 / 2 / 0, CR4 r2 671 / 0 / 0, CR5 370 / 0 / 0, CR6 243 / 0 / 0, CR7 496 / 0 / 0 and CR8 486 / 0 / 0 per build (F / K / X). Two limits apply, as for FlowX Platform MCP: the registered rubric counts a wiki TODO as a flag, so H14h is informative on CR3, CR4, CR5 and CR8 only; and "not refuted" does not say the rule was followed: it is the outcome column that does, and for CR6 and CR8 it does not. No Agentic Harness register mentions a change request.

On the latest versions (A28–A31, exploratory), CR6 r1–r3 are other, 0% on every set, F 0 / K 0 / X 243 each, nothing silent; CR7's scored builds followed the change (99.36 / 99.37 / 99.36 / 99.36 / 99.37 business, F 491 / K 0 / X 5); CR8 r1–r3 are other, 0% on every set, X 486 each, nothing silent. These are the outcome classes of the counted A25 builds; H14h is not re-tested.

Wiki ablation under disguise (A17, on the A20 version). Business equivalence, %; strict in brackets where it differs; per-set figures are the median build's.

Table 17. Wiki ablation under disguise, Agentic Harness (A20 version): business equivalence per arm, strict in brackets where it differs.

Arm	Builds (base)	Median of build means	base	c01	c02	c03	c04	Silent divergences
de-tailed	99.00 / 99.00 (49.00) / 99.00	99.004	99.00	99.00	98.99	99.00	99.03	last account in r1 and r3 (r2 flagged)
sum-mary	0 / 49.00 (49.00) / 49.00 (49.00)	48.812	49.00	49.00	49.49	49.00	47.57	r1: empty output, silent R-PASSTHRU; last account in r2 and r3; transaction ids flagged
no wiki	1.00 / 1.00 / 1.00	0.996	1.00	1.00	1.01	1.00	0.97	none (every difference flagged)
code	100 / 100 / 0	100.000	100	100	100	100	100	r3: no output, silent R-ABSENT

H12 supported, at the margin: code minus detailed is 0.996 points (c02 alone 1.01), the last-account record, as for FlowX Platform MCP. **H13 supported:** detailed minus summary is 50.2 points. The summary wiki gives no transaction-identifier rule; summary r2 and r3 number transactions from 1 and flag it, where the FlowX Platform MCP summary builds filled the rule from recall. **Instrument check passes** at 98.0 points. No Agentic Harness build names CardDemo or uses a recalled constant; the one possible signal (the legacy interest type code in no-wiki r2) was recorded before execution and can touch only one flagged transaction. For the Agentic Harness, which showed no sign of recall here, the ablation measures the wiki itself.

Four screens (A24, on the versions of A26–A31). The arm was registered under A26 and re-registered under A27–A31, always on three builds of its own and never pooled with FlowX Platform MCP. No Agentic Harness build built a screen. The counted builds are A30 r1 and A31 r2. Both are absent on every scored case, so H15h–H18h fail on two of three builds and are refuted as registered. Of the three registered builds, the two counted ones (A30 r1, A31 r2) produced no screens and are scored as absent under the registered rule; if those two builds are set aside, no verdict is reached. H19h was not run: no build has a UI flow, so no payment was made. The refutation records that nothing was built. It says nothing about how an application the Agentic Harness builds behaves against legacy, and no FlowX Platform MCP result can be read across to it. The "one wiki" limit stands: RQ3's wiki comparison does not extend to the screens.

4.10 Cost and time

[paper/COST_TIME.md]

A FlowX Platform MCP build of one step costs $2.29–5.44 in model usage at list price (median $3.46, 42 builds of the main, A5, A6 and A6b arms, across four steps) and takes 4.9–13.6 minutes (median 7.5). An Agentic Harness build takes 9.2–49.7 minutes (median 20.4, 24 builds) and 0.95–1.98M input tokens; the Agentic Harness does not log a dollar cost and we do not price it. Per thousand COBOL lines (comments included), the FlowX Platform MCP medians are:

Table 18. Cost and time per thousand COBOL lines, comments included: medians by step.

Step	Lines	FlowX Platform MCP $/kLOC	FlowX Platform MCP min/kLOC	Agentic Harness input tokens/kLOC	Agentic Harness min/kLOC
INTCALC	652	4.56	12.0	2.13M	31.9
POSTTRAN	731	5.37	11.6	1.99M	24.8
CREAST-MT	1,154 (CBSTM03A + CBSTM03B)	3.49	7.7	1.24M	18.6
TRANREPT	649	4.65	10.7	2.15M	29.8

The largest program has the lowest cost per line, so these ratios do not scale linearly with program size; four programs of 649–1,154 lines cannot show how they would scale. The figures exclude generating the wiki, human and driver time, and running the built workflows.

These figures predate the change request and the ablation. Those 18 FlowX Platform MCP builds cost $2.25–4.07 each for the change request (6 builds, $18.32 in all) and $1.56–3.29 for the ablation (12 builds, $28.80 in all) [runs/a16/SUMMARY.md, runs/a17/SUMMARY.md].

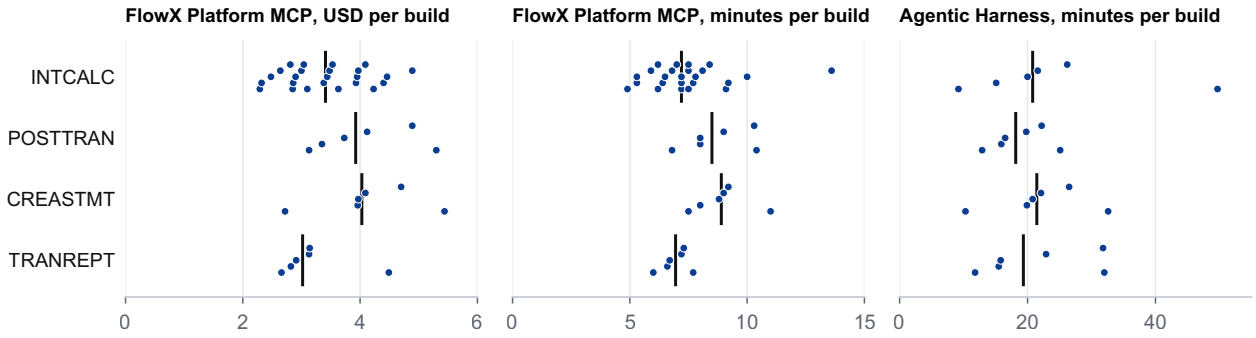

Figure 12. Cost and time per build, from paper/COST_TIME.md, which reads them off the run logs. Each dot is a build, the tick is the step's median. FlowX Platform MCP: n = 42, the main, A5, A6 and A6b arms; later arms fall outside this set (§4.10). Agentic Harness: n = 24, first scored run. The Agentic Harness does not log a dollar cost and is not priced. Excludes generating the wiki, human and driver time, and running the workflows. Axes start at zero.

4.11 How the FlowX Platform MCP builds ended

[runs/*_SUMMARY.md, runs/a16/SUMMARY.md, runs/a17/SUMMARY.md, runs/full/A32_RESULTS.md, runs/a21/interventions.log, runs/a24/interventions.log, runs/full/interventions.log]

The FlowX Platform MCP batch arms had no stop and no lost build; all 18 FlowX Platform MCP builds of the six further change requests ended with no stop and no retry, and all 18 returned output. One build, A6b Opus thin r3, was restarted after a sign-in expiry, before its builder had run [notebook §34]. On the screens one FlowX Platform MCP session was cut off by a usage limit (§6.2). On the full application, all three FlowX Platform MCP builds completed with no stop, retry or resume. Infrastructure interventions (sign-in refreshes, a client connection that was reopened, operator pauses between builds, concurrent non-study activity logged with

its start and end, and one diagnostic run that stopped on an expired sign-in before any workflow ran) are logged with times; none changed a build's inputs or prompt. Every lost build of either builder is scored as it stands; none is dropped.

5 Findings

F1. A change made in the wiki alone was carried into every FlowX Platform MCP build, against memory. All six FlowX Platform MCP change-request builds implement the edited rule and match the edited legacy program: 100% on CR2, for which we know of no public source, and 98.99–99.03% on CR1, whose only divergence is the legacy last-account defect (§4.1). CR2 r1 and CR1 r1 recalled the public program's old rule and said so in their registers, CR2 r2 called the rule the one most likely to differ from legacy, and all three still followed the wiki. Here, a stated rule took precedence over recall, the reverse of F3, where recall filled a rule the wiki left out. On the Agentic Harness, all six builds put the edited rule in their code; CR1 matches FlowX Platform MCP in 3 of 3 builds, CR2 matches the edited legacy on every record in 1 of 3, and no Agentic Harness build kept the old rule (§4.9). On six further rules, across all four steps, no FlowX Platform MCP build kept a legacy rule and 16 of 18 followed the change (§4.1). On the Agentic Harness, no build of the six rules kept a legacy rule silently: 10 of 18 followed the change at record level, including 3 of 3 on CR3, CR5 and CR7. On CR6 and CR8 all six builds are "other" and 0% at record level, because every statement differs on a flagged address line; only a post-hoc field-level count shows the rule followed there (§4.9).

F2. Under disguise, the detailed wiki is within a record of the code, and the rest is recall. With CardDemo disguised, detailed-wiki builds score within 1 point of builds given the COBOL (0.996 points on the pre-registered figure, 0.01 over the limit on one set), the difference being the legacy last-account defect, which the code arm copies and the wiki arm does not (§4.2). The summary arm matches the detailed arm in business terms in two of three builds, so for this step the business score did not need the Business Rules section; the summary builds' text fields come from recall. With no wiki the FlowX Platform MCP builds recognize CardDemo and fall back on recalled values that the disguise has changed, scoring about 1–4%. The Agentic Harness, whose ablation builds showed no sign of recall, gives the same code-to-detailed gap and supports H13 by 50 points: without the Business Rules section it numbers transactions from 1, the rule the summary wiki leaves out (§4.9). For a builder that does not remember the code, the detailed wiki is what carries the result.

F3. When the documentation omits a rule, the score measures the model's prior knowledge. The thin Opus wiki omits the fallback and states a different formula. The FlowX Platform MCP builder filled both from memory of the public source and scored 99–100%; the disguise turned the same memory into about 1%. The Agentic Harness builder showed no sign of such recall: it built what the thin wiki says, a different rate model. A private codebase gives no training-data recall to fall back on, so a thin wiki should be expected to perform as it did under disguise. POSTTRAN shows the same split: the FlowX Platform MCP build that chose the recalled legacy rules scores 100%, the builds that follow the thin wiki's own credit rule 72–87%. On layer 1 recall reaches the detailed wiki too. The Opus-wiki TRANREPT builds score 100% by recalling CBTRN03C's end-of-file behavior, and one Opus-wiki CREASTMT build reproduces the excluded overflow statement from the recalled table size. On the Sonnet wiki, which gives no rule for address line 3 or the page size, two FlowX Platform MCP CREASTMT builds cite CBSTM03A's `STRING … DELIMITED BY` for the address and all six FlowX Platform MCP TRANREPT builds take the 20-line page from recall; the Agentic Harness builds, which showed no sign of recall, differ from legacy on address line 3 in every statement and print 7 pages for 18. Where a wiki is silent, a match with legacy on a public codebase is not evidence for the wiki.

F4. Most builds that differed from legacy said so in advance. Of the five undisguised FlowX Platform MCP builds that differ on the last account, four flagged it before scoring: two by reasoning about the wiki's wording, two from recall. The assumption registers made it possible to map flags before any output was opened. The

Agentic Harness builds kept detailed registers too (15–61 entries) and recorded several rules as unspecified by the wiki. On POSTTRAN no build that returned records has a silent divergence; Agentic Harness Opus r1 and r3 recorded the reject codes and the cycle-total sign as unspecified. Layer 1 holds the same pattern: no build that returned output differs silently. The FlowX Platform MCP CREASTMT builds that print the six extra `c03` statements, and the FlowX Platform MCP Sonnet TRANREPT builds that decline to copy the doubled last amount, each recorded the choice and its departure from legacy before scoring. Flags are coarse, though: FlowX Platform MCP builds raised 16–26 flags per layer 1 build, none of them needed on CREASTMT `base`. The record weakens under disguise. Among the FlowX Platform MCP builds of the main CardDemo arms that returned output, one divergence was silent (INTCALC FlowX Platform MCP Sonnet r1); the same last-account divergence is silent in the three FlowX Platform MCP CR1 builds, in FlowX Platform MCP CR3 r2 and r3, and in one disguised detailed build of the ablation. The registered screens test has silent units too (§6.2). On the disguised INTCALC (A5), four of six builds had a silent divergence, and Sonnet-wiki r2, which flagged the last account undisguised, did not flag it disguised [`runs/A5_SUMMARY.md`]. Part of the flagged-in-advance record on public code comes from memory of that code, and F4 should be read as an upper bound until the honesty-rate analysis (§9) is run.

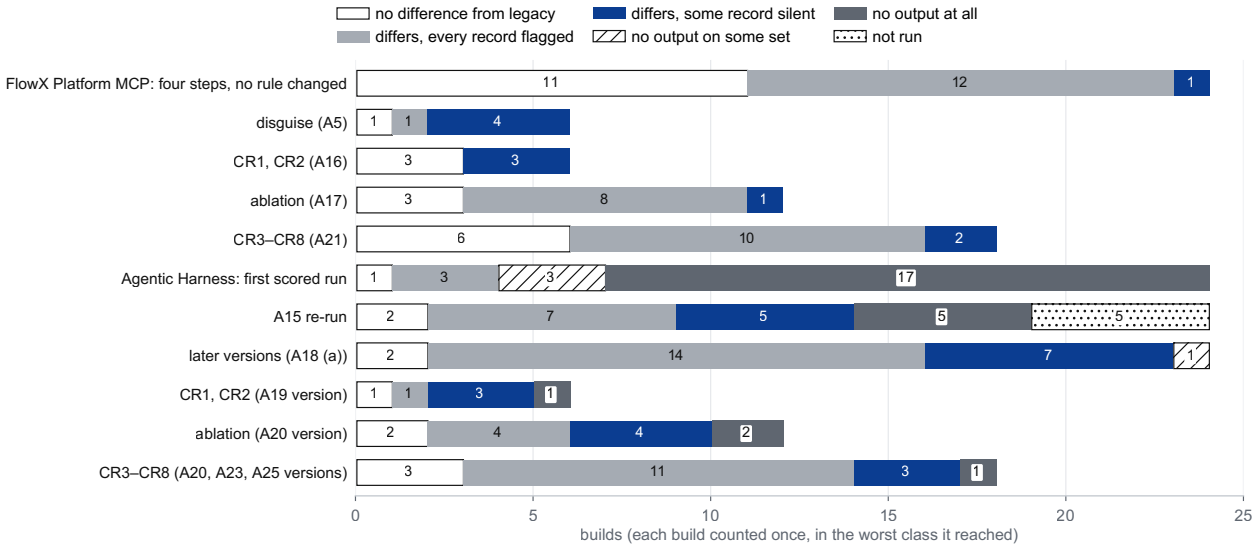

Figure 13. Whether builds said so in advance. Each build is counted once, from the silent and flagged counts in its report files; a build or set that returned nothing counts as silent and is drawn apart, except one build's empty lists, all flagged (POSTTRAN Opus r3, A15 re-run). Later versions: A18 (a), exploratory. Flags are coarse, and the flag mapper is the AI assistant that ran the experiment. Classes are derived post hoc for this figure from the registered silent and flagged counts; no hypothesis uses them. n per row = builds in that arm.

F5. On the whole application, the documentation, not the code, is what moved (exploratory). From the wiki alone, the full application did not match legacy (A32, §6.3). With a screen specification derived from the source and build rules, developed over a series of development builds in which only these written inputs changed and no build was repaired, the development run matched legacy on every one of the 54 screen cases on untagged units, every batch case and every end-to-end pair, and passed the browser check (§6.4). The inputs were tuned on these cases, so this is an observation, not a test; the confirmatory test (§6.5) is the test. There, two of three fresh builds passed every condition and the third missed on two units of one screen, so full equivalence is not claimed.

6 Beyond batch: a UI pilot, a registered test of the screens, the full application, a development run and a confirmatory test

6.1 The pilot (unregistered, not compared with legacy)

[runs/a21/interventions.log]

Status. This is a pilot outside the pre-registered study: no amendment, no hypothesis, no score against the legacy program. It is reported because it uses the same wikis beyond the batch steps, and because the registered test (§6.2) is read beside it. It is not part of the abstract, the findings or any figure in the summary.

What was built. On 2026-10-29, on the same development environment (platform 5.13), the FlowX Platform MCP builder (`claude-opus-5-5`, FlowX Platform MCP and file tools only) built four CardDemo screens from wiki pages and seed files: sign-on, the main menu, account view and bill payment, with a FlowX.AI Database data layer (five stores, loaded from the study's shipped data and CardDemo's sample sign-on users) and workflows behind the screens. It ran in its own project in another workspace, on a separate builder profile, with start and end times logged beside the study's builds; no Agentic Harness build ran in its windows. Three builder sessions: the first two screens (12 min 45 s, USD 10.70), one fix session given two browser screenshots (7 min 40 s, USD 3.13), and the menu and bill payment (17 min 43 s, USD 8.68): about 38 builder-minutes and **USD 22.51** in all, excluding operator and check time.

Checks (our own scripts, independent of the builder's tests; against CardDemo's data, not against the legacy screens):

- *Workflow level:* five accounts × nine displayed fields, 45 of 45 equal to the seed data; sign-on accepts a valid user and rejects a wrong password and an unknown user with the wiki's message; unknown and non-numeric account ids get their own messages; no response contains a password. Bill payment on one account: 15 of 15 checks, including the confirm values Y, N and an invalid one, a payment record of type 02 and category 0002 for the full balance with the next transaction id, the balance set to 0.00, exactly one transaction added, and "nothing to pay" on a second payment. The shipped data has no account with a balance at or below zero, so that message was reached only by paying twice.
- *Browser:* a scripted browser run on the live screens. The first run found two failures (a wrong password cleared the form with no message; dates shown in a short format, so 6 of 9 fields matched per account). After the one fix session the second run passed every step, and the third, after the menu and bill payment were added, passed all 12 steps with 45 of 45 account fields. Not covered in the browser: a typed menu option, the admin sign-on, Exit, Clear, and the N and invalid confirm values on bill payment (the last two were checked at workflow level).

Mixed wikis, disclosed. Sign-on and account view were built from Opus-wiki pages; the menu and bill payment from Sonnet-wiki pages, because the Opus-wiki bill-payment pages describe a user-entered partial amount and leave writing the transaction as a TODO. The project data model page stayed the Opus wiki's. Each bundle also carried a contract that set the pilot's scope and routing and, after the fix session, a form-handling pattern drawn from the pilot's own browser failures. The pilot therefore tests neither wiki alone, and its builder was helped by what the first browser run showed.

Deviations from the wiki. Sign-on checks seeded users where the wiki authenticates through the identity provider. Admin users land on the main menu, because the admin menu was not built. There are no BPMN processes: the screens call workflows directly, where the wiki chose a process with one call per menu option. One menu option is hidden by a constant where the wiki has a feature-toggle type. Bill payment makes two ordered writes with a compensating restore where the wiki asks for one logical unit (an open decision in the wiki). The payment's merchant city and zip are `N/A` where the wiki leaves the literals as a TODO. The confirm

field is a one-character input, not a Y/N select. The role matrix and the lockout are not enforced. One account-id rule ("not all zeros") came from the builder's knowledge of the legacy code, not from the wiki. The transaction seed is the posting batch's output while the account balances are the pre-batch values, which does not affect these screens.

A platform convention: the date format. The account-view wireframe shows dates as YYYY-MM-DD. By platform convention the web text component shows an ISO date string in the project's short date format, and a build that needs another format sets it on the component; setting it through the build tools was not tested. The builder returned the dates with non-breaking hyphens (U+2011), which look the same on screen, but copied text and screen readers get U+2011, not "-". We record this as the build missing a display setting the convention requires, not as a tested cause.

What the pilot does not show. It is not compared with the legacy screens: the online oracle (§3.7) was not used, and no screen output was scored. One session used a fix informed by screenshots. The pilot's form pattern and contract are excluded from the inputs of the registered test.

6.2 Registered test against the online oracle (A24, FlowX Platform MCP arm)

[PROTOCOL A24, `prereg/frozen/a24/`]

Design. A24 was frozen and stamped before any build. It uses one wiki, the Sonnet wiki (a bundle of 26 pages and a contract), and fresh projects; no build used the pilot or was told of it, and each transcript was checked for access to the pilot project (none). FlowX Platform MCP built r1–r3. Each build ran the 17 online cases (16 scored) at workflow level against `golden-online/v1`, a browser subset of four cases, and one end-to-end case: a bill payment made on the FlowX screens and carried through the FlowX batch builds, with its effect (a chain with the payment against one without) compared with the same payment's effect on the legacy chain. A hypothesis is refuted for a builder if two or more of its three builds fail it. Units that rest on a stub rule are tagged and reported apart, because the online oracle has no reference CICS and is weaker evidence than the batch oracle.

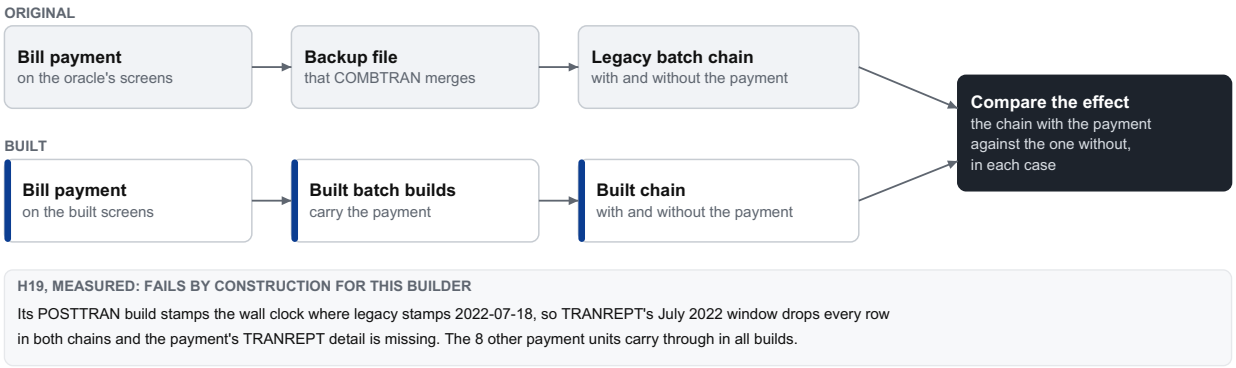

Figure 14. The online-to-batch path of the end-to-end case in A24: the payment's effect on the built chain is compared with its effect on the legacy chain. Diagram: no data plotted.

The target claim is not stated. A24 registered a claim covering batch through to working screens and allows it for a builder only if H15–H19 all hold. For FlowX Platform MCP, H16–H19 are refuted. What the data supports is narrower: screens were built and scored against the original, the payment's file state holds on the two registered payment cases (a third, tagged case differs in two builds, unflagged in one), and the screens differ as set out below.

Verdicts as registered (r1, r2 as it stands, r3).

Table 19. A24 verdicts as registered, FlowX Platform MCP (r1, r2 as it stands, r3).

Hypothesis	r1	r2	r3	Verdict
H15 payment files, silent units	0, holds	0, holds	0, holds	not refuted (0 of 3 fail)
H16 core screens, silent units	23, fails	171, fails	113, fails	refuted (3 of 3)
H17 account view, fields equal	16 of 29, fails	12 of 29, fails	12 of 29, fails	refuted (3 of 3)
H18 browser subset, steps	24 of 25, fails	24 of 25, fails	24 of 25, fails	refuted (3 of 3)
H19 end to end	payment differs, fails	payment differs, fails	payment differs, fails	refuted (3 of 3)

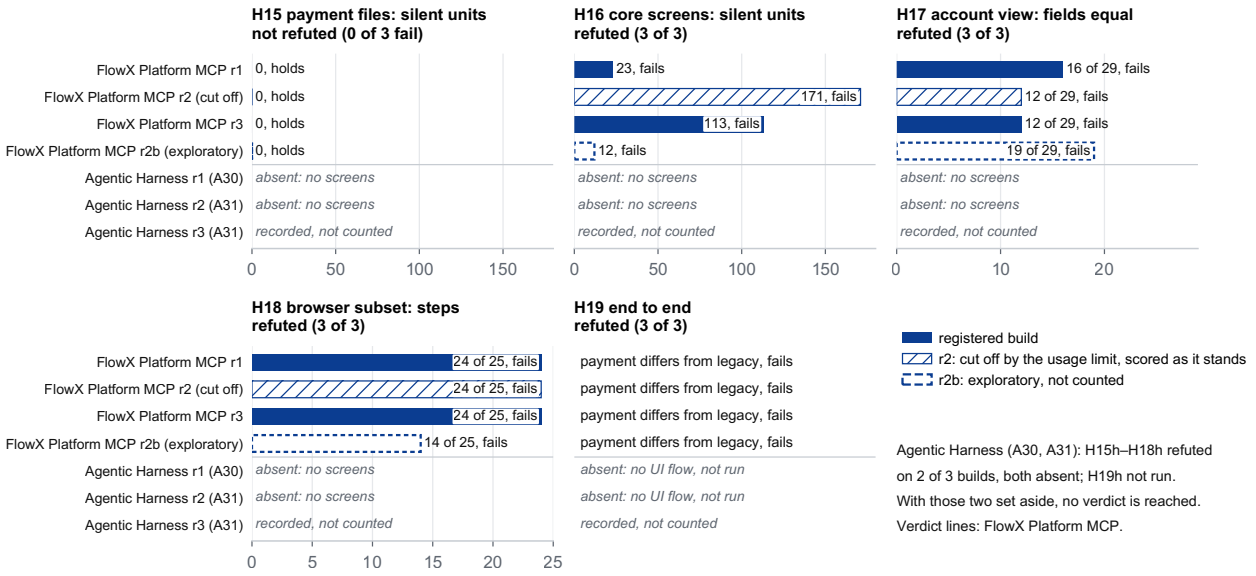

Figure 15. A24 screens, per hypothesis and build, scales differing per panel. FlowX Platform MCP: n = 3 registered builds; r2 was cut off by the account's usage limit and is scored as it stands, with no flags, so every difference in it counts as silent; r2b was added after the limit reset, is exploratory and post hoc, and is never counted (10 of its 11 failed browser steps come from the operator's screen map, not the build). H15 covers the two registered payment cases; a third, tagged payment case differs in r2 and r3 (next figure). Agentic Harness (A30, A31): the counted builds are A30 r1 and A31 r2; both produced no screens and are scored absent under the registered rule, so H15h–H18h are refuted on 2 of 3 as registered; if those two builds are set aside, no verdict is reached. A31 r3 is recorded, not counted. H19h was not run: no build has a UI flow. Earlier versions (A26–A29) are not plotted.

Descriptive columns (no threshold). r2b is exploratory and is never counted.

Table 20. A24 descriptive columns, no threshold. r2b is exploratory and is never counted.

	r1	r2	r3	r2b (exploratory, not counted)
Screen units equal, untagged	186/547	178/533	177/533	369/559
Screen units equal, tagged	58/200	54/172	55/172	54/221
File units equal, untagged	6747/6747	6747/6747	6747/6747	6747/6747
File units equal, tagged	8/8	6/8	6/8	6/8
Flags	82	0	93	79
Post hoc, blank extras excluded: H16 silent units	3	121	93	3
Post hoc: screen units equal, untagged	186/435	178/435	177/435	369/435
Post hoc: screen units equal, tagged	58/88	54/83	55/83	54/90
Cost	$10.25	$12.04	$13.64	$10.83
Builder wall clock	18 min	24 min	24 min	34 min

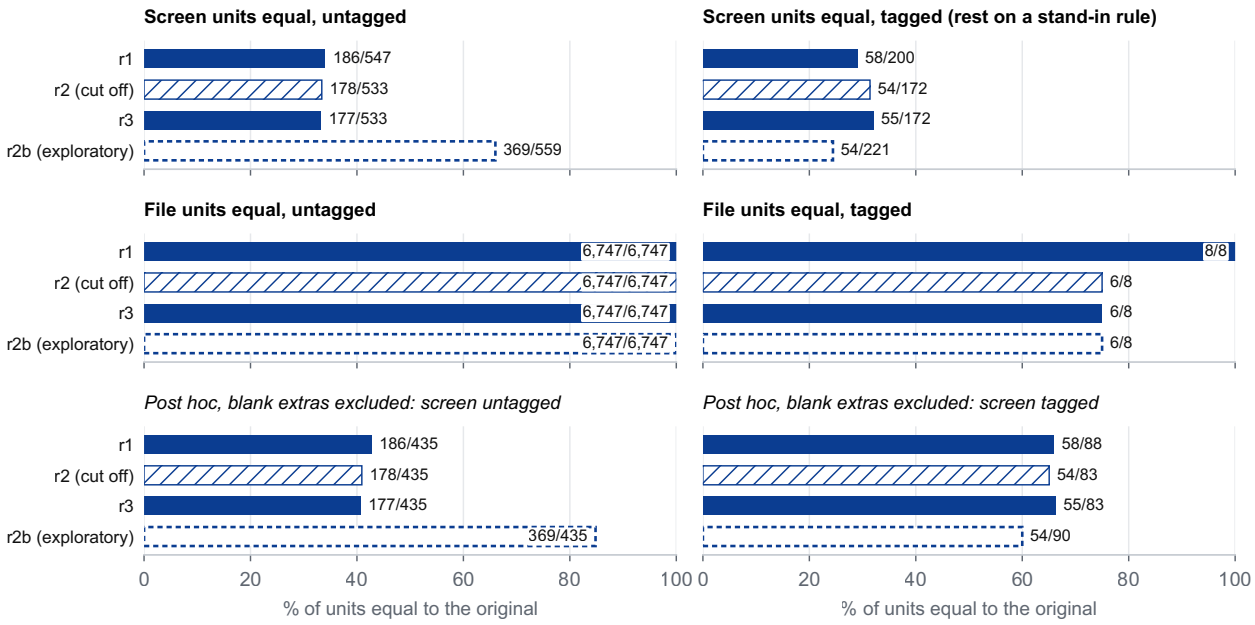

Figure 16. A24 screens, descriptive measures with no threshold: units equal to the original, FlowX Platform MCP. n = 3 registered builds plus r2b, exploratory and never counted. r2 is scored as it stands. Tagged units rest on a stub rule and are weaker evidence. The two lower panels are post hoc, exclude blank extras, and do not replace the frozen numbers. No Agentic Harness build is plotted: no build of it has built a screen.

r2 was cut off. The builder session ended at the account's 7-day usage limit after 363 tool calls. This was not a builder decision. r2 is scored as it stands under the frozen rule: its latest version, its own temporary test-data operations still in the project, and 0 flags, because the session ended before it wrote a register. So every difference in r2 counts as silent, which raises its H16 count. **r2b is exploratory:** a build with the same prompt, model and instrument, added after the limit reset, labelled post hoc, reported beside r2 and never in its place. With r2b in r2's place the verdicts do not change: H15 not refuted (0 silent units); H16, H17, H18 and H19 refuted (12 silent units; 19 of 29 fields; 14 of 25 steps; payment differs).

r2b's browser figure is 14 of 25, as recorded, and is not a measure of the page. 10 of its 11 failed steps come from the operator's screen map, not from the build: the map's detection order read the bill-payment card as the menu, and the check went on from there (o-bil-01 and o-bil-03, steps 3–7). The map was committed before any output was opened and is unchanged. No corrected run was made. The eleventh failure is the same step as in r1–r3.

What the failures are. Each cause is labelled as measured or inferred.

- **H16. Measured.** Two kinds of silent difference make up nearly all of it. The menu's option labels differ from legacy in r2 and r3 ("1. Account View" where legacy shows "01. Account View"). Blank inputs that the legacy step does not send are the rest: they account for 20 of r1's 23 units, 50 of r2's 171 and 20 of r3's 113.
- **H17. Measured.** The fields that differ have no separate value the operator's screen map could read, mostly name and address (9 fields in every build), and the SSN is masked in every build. r2's differences are silent because r2 has no flags; in r1, r3 and r2b they are under flags.
- **H18. Measured:** all three registered builds, and r2b, fail at the same step, o-avw-01 step 4, on the same four date fields (ADTOPEN, AEXPDT, AREISDT, ACSTDOB), where the page shows a value that differs from what the workflow returns (in r1, M/D/YY). **Inferred, not tested:** that the build missed the date display setting the platform's convention requires (§6.1). The inference rests on the same fields failing on every build.

- **H19. Measured:** the failure is by construction for this builder. Its POSTTRAN build stamps the wall clock where legacy stamps 2022-07-18, so TRANREPT's July 2022 window drops every row in both chains, and the payment's own TRANREPT detail is missing. The verdict is reported as frozen. The 8 other payment units (combtran, CREASTMT ×5, INTCALC and POSTTRAN `curr_bal`) carry through in all builds. TRANREPT was not rerun with other dates, because that would change frozen inputs, so what it would show with a matching window is not measured.

Instrument fixes, disclosed. Fixes 1–4 were found on the adapter check on the pilot app, never scored, before the freeze; fixes 5 and 6 came after it.

1. Operator loader: built against the live tools.
2. Replay: nested start parameters.
3. Adapter check: amounts compared as decimals.
4. Browser check: clears inputs the browser profile autofilled, and reads input values.
5. Report scorer: it stopped on a blank TRANREPT total; a blank or missing total now parses as missing, and the frozen H19 rule applies as written. Self-tests pass, 77 of 77 layer 1 report sets re-score identical, and the end-to-end control passes 7 of 7.
6. Adapter: a show path "a|b|c" takes the one non-empty text among mutually exclusive page texts for one legacy field; two or more filled count as differing. The ruling was written before r2 was executed; it does not apply to r1, and r3 and r2b do not need it. Beside r2, with its error message read from one text only: H16 176 silent units, screen units 177/533 untagged and 47/172 tagged; H15, H17 and the case outcomes are unchanged.

Post-hoc blank-input column. The online scorer's self-test could not catch blank extras, because its candidates are exported from the oracle and never carry blank fields a step does not send. The "post hoc" rows above exclude them. That exclusion is not part of the frozen rule, and those rows do not replace the frozen numbers.

Other disclosures. A payment unit whose whole TRANREPT unit is missing in both chains is keyed `missing`, so it counts as not differing (instrument property, unchanged). r3's browser subset ran after a browser sign-in; earlier attempts stopped before any case ran. r3 and r2b ran on usage overage, with rate-limit events and no interruption. r2b's execution was restarted before any case had started. One context call in r2b's execution went to the wrong builder profile and was restored at once, with no workflow run in between. In r2, the account view's update of the session account id is not carried by the operator's map. The transcripts contain sample passwords the builders used as test data; these are masked, and they may be a sign of recall.

The Agentic Harness arm of this test (A26–A31) is reported in §4.9: no screen was built, H15h–H18h are refuted on two of three counted builds, and H19h was not run. Of the three registered builds, the two counted ones (A30 r1, A31 r2) produced no screens and are scored as absent under the registered rule; if those two builds are set aside, no verdict is reached.

6.3 The full base application (A32, FlowX Platform MCP arm)

[PROTOCOL A32, `prereg/frozen/a32/` (`FULL_A32.md`, published as the reading copy `FULL_A32.REDACTED.md`, binds where the entry is silent); `runs/full/A32_RESULTS.md`]

Design. A32 measures the rest of the base application as A24 measured four screens and A1–A21 the four core batch steps: 17 online programs (4 of them scored in A24), the date utility CSUTLDTC, and 11 batch programs (the 4 core steps and 7 others). The optional Db2, IMS and MQ modules stay out. The answer keys were all frozen before the amendment and none was built against: `golden/v1`, `golden/full-v1`, `golden-online/v1` and `golden-online/v2`, giving 54 scored screen cases, 23 batch cases and 4 end-to-end pairs, plus 9 browser cases. One builder: the Agentic Harness is not part of A32. FlowX Platform MCP built r1–r3

from one bundle (the Sonnet wiki of the whole application, 234 pages), `claude-opus-5-5`, one session each, in fresh projects. Each launch excluded the operator's global instruction file, and each session's load record shows it was not loaded. The target claim, batch to working screens, both checked against the original, is stated for a builder only if HF1–HF7 all hold. Each hypothesis is refuted if at least two of three builds fail it, and each is decided on units not resting on R-CEE-PIC8, the oracle's reading of the vendor's date service (7 cases, reported in their own column).

Three contract additions, which make A32 a different test from A24. A `job_requests` store; a run date (`RUN_TIMESTAMP = 2022-07-18 10:00:00`), created by the operator as a project parameter before each builder started, and the path a workflow script reads it from; and a text on dates on screens. A24's contract carried none of these. So HF5 is not A24's H18, HF6 is not A24's H19, and A32's results are never shown as an improvement on A24's.

Builds. All three ended on their own, with no cut-off or resume: r1 69 min, $37.67; r2 75 min, $35.62; r3 74 min, $35.77 ($109.06 in all, list price). The authors approved r2 and r3 on r1's cost before any r1 output was executed or scored, as A32 requires.

Verdicts as registered.

Table 21. A32 verdicts as registered, FlowX Platform MCP, full base application (r1–r3).

Hypothesis	r1	r2	r3	Verdict
HF1 all 17 screens built and reachable	fails: 4 not reachable, 2 differ	fails: 2 not reachable, 1 differs	fails: 3 not reachable, 1 differs	refuted (3 of 3)
HF2 screens differ only where flagged	fails: 0 of 17 programs with no silent difference	fails: 1 of 17	fails: 1 of 17	refuted (3 of 3)
HF3 money files written as legacy	holds: 0 silent units	holds: 0	holds: 0 (1 more resting on R-CEE-PIC8, reported apart)	**holds (0 of 3 fail)**
HF4 list paging returns legacy's rows	fails: 336 of 348 units equal	fails: 336 of 348	fails: 336 of 348	refuted (3 of 3)
HF5 browser shows what the workflow returns	fails: 5 of 9 cases as legacy (44 of 60 steps)	fails: 8 of 9 (57 of 60)	fails: 1 of 9 (37 of 60)	refuted (3 of 3)
HF6 a session carries through the batch chain	fails: all 4 absent	fails: 2 of 4 carry	fails: 2 of 4 carry	refuted (3 of 3)
HF7 export and import round trip	fails	fails	fails	refuted (3 of 3)

● holds on this build ○ fails on this build

		r1	r2	r3	verdict
HF1	all 17 screens built and reachable	○ 11/17 as legacy	○ 14/17 as legacy	○ 13/17 as legacy	refuted 3 of 3
HF2	core screens differ only where flagged	○ 0/17 clean	○ 1/17 clean	○ 1/17 clean	refuted 3 of 3
HF3	money files written as legacy	● 0 silent	● 0 silent	● 0 silent (+1 PIC8)	**holds 3 of 3**
HF4	list paging returns legacy's rows	○ 336/348 equal	○ 336/348 equal	○ 336/348 equal	refuted 3 of 3
HF5	browser shows what the workflow returns	○ 5/9 cases	○ 8/9 cases	○ 1/9 cases	refuted 3 of 3
HF6	a session carries through the batch chain	○ 4/4 absent	○ 2/4 carried	○ 2/4 carried	refuted 3 of 3
HF7	export and import round trip	○ differs	○ differs	○ differs	refuted 3 of 3

Figure 17. The full base application (A32), FlowX Platform MCP: HF1–HF7 per build, as registered; n = 3 builds, each scored on 54 screen cases, 9 browser cases, 23 batch cases and 4 end-to-end pairs. A filled dot holds, a ring fails. Cells: HF1 programs reached as legacy; HF2 programs with no silent screen difference; HF3 brackets the units resting on R-CEE-PIC8, which decide nothing. A hypothesis is refuted if at least 2 of 3 builds fail. Only HF3 holds. HF5 is the registered browser run; the post-hoc fix 11 rerun (§6.3) is not plotted. Units resting on R-CEE-PIC8 decide nothing and are counted apart. A32's contract adds a store, a run date and a date-display text that A24 did not carry, so HF5 and HF6 are not A24's H18 and H19. Not post hoc.

Descriptive columns (no threshold). Screen units equal, untagged: r1 1,855/3,278, r2 1,928/3,254, r3 2,102/3,262. File units equal, untagged: 25,493/25,493, 25,488/25,492 and 25,488/25,493. R-CEE-PIC8 column: 2,779, 2,768 and 2,770 of 3,045. Per program, the silent screen units run from 0 (COSGN00C in r2 and r3) to 140 (COUSR00C in r1); the batch programs and the core steps are in `runs/full/A32_RESULTS.md`. In r2 and r3 the core batch steps score close to before: POSTTRAN 100%, INTCALC 98.99–99.03% (the last account, silent), CREASTMT 100% on four sets and 86.27% on c03 (flagged in r2, silent in r3), TRANREPT 99.72–99.73% in r2 (one flagged unit per set) and 99.45–99.46% in r3 (two silent units per set). The export and import round trip matched 414–416 of 483 and 464 of 477 units.

What the failures are. Each cause is labelled as measured or inferred.

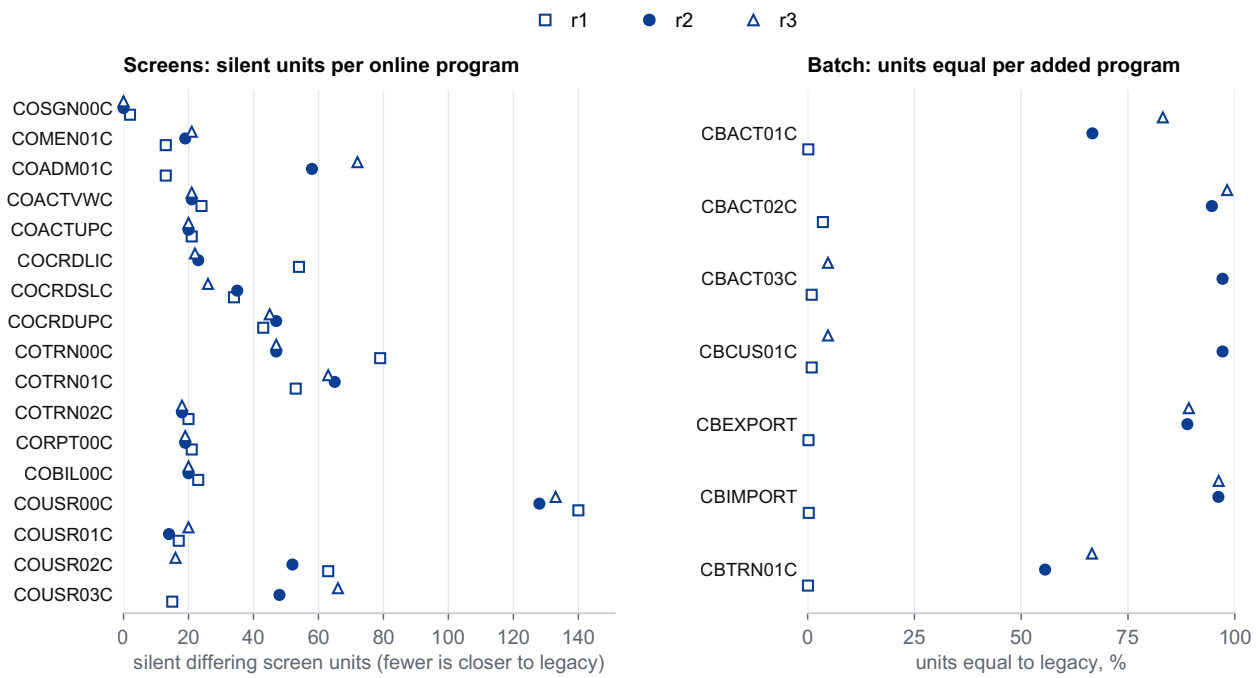

Figure 18. A32 per program, FlowX Platform MCP, n = 3 builds; markers tell the builds apart. Left: silent differing screen units per online program on its scored cases (untagged; 0 is no silent difference). Right: units equal to legacy for the seven added batch programs (CBACT01C–03C, CBCUS01C and CBTRN01C have no behaviour in the wiki and carry no hypothesis, D6; CBEXPORT and CBIMPORT are the HF7 round trip). r1's batch outputs are empty because its end nodes pass nothing on, so it sits near 0. Descriptive; no threshold.

- **HF1. Measured.** COCRDUPC is not reachable in any build, COTRN01C in r2 and r3, COMEN01C in r1 and r3, and COADM01C and COUSR02C in r1. The account update (COACTUPC) is reached in every build but its core case differs silently.
- **HF5. Measured.** r2 fails one case on navigation (the user list returns to a different screen than legacy). r1's list pages show no rows although its workflows return them, and r1 has no button for the admin menu's option 5. r3's failures are mostly the instrument's (below); its own are a goodbye page where legacy shows the sign-on screen (two cases) and the same user-list navigation as r2.
- **HF6, HF7. Measured.** r1's batch workflows compute their outputs in a script node and pass none of them on at the end node, so every r1 batch output is absent, which is build behaviour and is scored that way. In r2 and r3 the transaction add carries through except what it inherits from upstream differences, the report request carries through, and the account update and the second report request differ from legacy.

Instrument fixes, disclosed. All were made after the freeze and before any output of the build they first served was executed, each with a self-test and negative controls; answer keys, cases and scorers are unchanged. Fix 7 splices legacy fields into one web box and back (dates, SSN, phones, card expiry); fix 8 maps several legacy mark fields to one build control, and keeps a value an action's form mapping fills across that action's screen change; fix 9 adds conditional actions by page mode, the selected list row carried to the next page, and browser handling of menu buttons, radio and dropdown choices and screen detection by card title; fix 10 sends a constant START parameter a build's own workflow needs to select the program. The browser check ran in the operator's signed-in Designer browser and used only the tabs it opened. The run date is an operator step, logged as an intervention in each build.

Post-hoc browser sensitivity (fix 11), decides nothing. r3's registered browser run showed two instrument gaps: its menus are numbered lists whose numbers the page shows but the text reader drops, and its read-only dropdown did not take a plain click. Fix 11, off by default and self-tested on local pages, reads the numbers and

opens the dropdown through its wrapper or the keyboard. Rerun on the same plans: r1 44/60 and r2 57/60 steps, unchanged; r3 from 37/60 to 54/60 steps and from 1 to 5 of 9 cases. HF5 stays refuted on every build. One r3 step still fails on the instrument (the map counts input boxes without the dropdown, so a confirmation lands in the wrong box). The registered result stands [`runs/full/A32_RESULTS.md`].

Other disclosures. The builds' transcripts hold the seed's sample password, masked before any commit; one map agent's debug output printed it in its own session and wrote it to no file. The first r2 loader attempt ran against r1's project context and created nothing; the execution script now sets and checks each build's project before every step.

6.4 A screen specification and build rules: the development run (exploratory, FlowX Platform MCP)

[inputs `builds/inputs/screen-spec/`, `scorer/CONTRACT-UI-FULL.md`, `scorer/CONTRACT-BATCH-FULL.md`]

Status. Exploratory. No amendment governs it, it has no hypothesis, and it decides nothing. It is reported because the confirmatory test (§6.5) freezes its inputs, and because it bears on the report's thesis. It is never pooled with A32, A24 or the confirmatory test.

What was added to the inputs. A32 showed where the wiki of the whole application was too thin to rebuild the screens. We added two written inputs, and nothing else.

- *A screen specification derived from the COBOL source* (`builds/inputs/screen-spec/`). One page per online program (17), stating what the legacy screen shows and does: layout, labels, field widths, messages, keys, navigation, validation order, paging and file effects. One page per batch program whose printed lines the wiki leaves out (8, with a shared page of file-status and abend rules). A catalogue of legacy defects (LD-01 to LD-150), each of which a build may reproduce or flag, and a list of places where the wiki and the source disagree. Every statement cites the COBOL source or its maps by file and line. The pages take precedence over the wiki for screens; the wiki stays the source for everything else.
- *Build rules* in the contracts. Some state legacy behaviour precisely (the target is every scored unit, except catalogued defects that the build reproduces or names; a step that stops through the abend service writes return code 16; indexed inputs are read in key order). Others are platform conventions a build must follow: the display options each UI component declares, how a node's output joins the workflow data, how a UI action passes values back, and how a plain-text screen is shown. The rest are checks the build makes on its own work and records before it finishes: reading the flow back, running each batch workflow on its inputs and with one input absent, reaching each screen and pressing the next key, and listing the keys each screen offers against the specification.

How the inputs were developed. The screen specification and the build rules were developed over a series of development builds and then frozen before the confirmatory test. Each development build was a fresh FlowX Platform MCP session in a fresh project, with the same model, prompt and tools as A32, and was scored with the instrument described below. Between builds only the written inputs changed. Each change was traced to the COBOL source or the screen specification (file and line), or to a platform convention seen in a build; none was taken from an expected output or a case. No built application was edited by hand. The inputs were developed against the same 54 screen cases, 23 batch cases and 4 end-to-end pairs that score them, so the development run is in-sample: its scores show that the inputs can carry a build to the legacy result on these cases, not that they will. Every change and its source are in the records of the development run (§8).

The development run. The last development build, the first on the frozen inputs (the bundle of 237 pages plus the screen specification), `claude-opus-5-5`, one session, ended on its own with no stop or resume: 995 turns, 622 MCP calls, none outside its project, 2 h 11 min, $67.96 at list price. Scored as A32, with instrument

fixes 11–18 on (§7.1). The run date was set by the operator before the builder started, as in A32. Infrastructure interventions during the development builds (sign-in refreshes, signing in again to the Designer browser, browser runs set aside unchanged and rerun because no case had started, and one browser run restarted after it stopped during a store reset, before any case had started) are logged with times; none changed a build's inputs.

Table 22. The development run (exploratory) against A32, full base application; A32 shown for reference only.

Measure	Development run	A32, for reference (r1 / r2 / r3)
Online cases equal to legacy, untagged units	54 of 54	not reported in this form
Screen units equal, untagged	3,147/3,147	1,855/3,278 / 1,928/3,254 / 2,102/3,262
Screen units equal, tagged	4,624/4,627	not reported in §6.3
Programs reachable	17 of 17	13 / 15 / 14 of 17 reachable
Programs with no silent screen difference, untagged	17 of 17	0 / 1 / 1 of 17
Money files, silent units	0	0 / 0 / 0
List paging units equal	348/348	336/348 each
File units equal, untagged / tagged	25,493/25,493 / 13/13	see §6.3
R-CEE-PIC8 column (decides nothing)	3,004/3,004	2,779 / 2,768 / 2,770 of 3,045
Batch cases matching legacy (added programs)	23 of 23 (8,023/8,023 units)	see `runs/full/A32_RESULTS.md`
Core steps on five sets, strict	20 of 20 at 100%	see §6.3
Export and import round trip	holds	fails ×3
End-to-end pairs carried through	4 of 4	0 / 2 / 2 of 4
Browser steps, cases	60/60, 9 of 9	44/60, 57/60, 37/60
Cost, time	$67.96, 2 h 11 min	$37.67 / $35.62 / $35.77; 69 / 75 / 74 min

A32's builds came from the wiki alone, under different contract additions; the column is shown for reference and is not an improvement measured under one design.

What still differs. Three tagged screen units, all unflagged:

- Two ERRMSG units on the sign-on screen (bill payment and admin entered without a communication area, catalogued defect LD-147). Legacy reaches this message when a program is started directly; neither the browser check nor the workflow adapter can start a program other than through the sign-on page, so no build can show it here. The confirmatory test treats this as an instrument limit, fixed in advance.
- One ERRMSG unit on the account view (catalogued defect LD-21): the message matches legacy except the RESP2 digits, whose value comes from the legacy runtime (140 in the reference run) and is given by no input. The confirmatory test scores the rest of the message and reports these digits apart.

Not scored: the export file's timestamp text. Legacy writes two fraction digits followed by four X'00' bytes; the build writes six fraction digits. The batch scorer compares the date part only, and the export check reports the difference on every record of the three export cases.

What the development run does not show. That the inputs carry a fresh build to the legacy result: they were tuned on these cases, and one build is one draw. That the screens match a real CICS: the online oracle is our reading of the vendor's documentation (§3.7). Correctness: the measure is a match with the legacy run, defects included.

What it shows about the thesis. Between development builds the application was never repaired. Each build started from the written inputs, and each change to the screens' behaviour was made in those inputs: in the specification, where it stated a legacy rule from the source, or in the build rules, where it stated a platform convention or a check. The scores moved with the documentation. This is the report's thesis applied to the whole application, as an observation on one exploratory series; the confirmatory test is where it is tested.

6.5 The confirmatory test (pre-registered)

[pre-registration and results published with this report]

Status. The design below was frozen and timestamped (OpenTimestamps, hash only) on 16 November 2026, before the first build. The three builds ran that evening, one at a time, and were scored on 16–17 November.

The question. Can a coding agent working only through FlowX.AI MCP rebuild the whole CardDemo base application (17 online programs, the date utility CSUTLDTC, 11 batch programs) so that every scored unit matches the legacy run, apart from defects of the legacy source catalogued in advance?

Inputs, frozen at registration. The bundle (237 pages: the wiki of the whole application, the data contracts and the build rules), the screen specification (17 program pages, batch pages, the defect catalogue LD-01 to LD-150, the list of contradictions), the prompt and the resume prompt, each by its hash. The build script checks every hash before each build and refuses on a difference.

Builds. Three fresh builds, r1–r3, `claude-opus-5-5`, one headless session each, only the FlowX Platform MCP server and file tools, the operator's global instruction file excluded and its absence checked in the session's load record. The run date is set by the operator before the builder starts, as in A32. Builds run one at a time. A session stopped by a usage limit is resumed with the frozen resume prompt and counts, with its resumes reported; any other stop ends the build, which is scored as it stands. No build is repeated or replaced.

Answer keys, unchanged and never built against: `golden/v1`, `golden/full-v1`, `golden-online/v1` and `v2`: 54 scored screen cases, 23 batch cases, 4 end-to-end pairs, and 9 browser cases (60 steps), as A32.

Scoring, per build, as the development run. Extract, mask, assumptions check, map agent, flags, loader, the 54 online cases at workflow level, the 9 browser cases in the operator's signed-in Designer browser, batch (23 cases and the 4 core steps on five sets), end to end, export check, rescore. Instrument fixes 11–18 on. The build's maps, flags and notes are committed before any of its output is executed.

Catalogued defects: scope fixed before r1. A frozen file lists, for each catalogued defect, the scored units (case, step, field) at which it can show. It was drawn up from the catalogue and the case inputs and, disclosed, from where the development builds differed. A difference is excused only at a unit in the scope of a defect that the build's assumptions flag by its identifier as a deliberate deviation. A defect stated as reproduced, or not named, excuses nothing: if the build says it reproduces a defect, its units must equal legacy.

Instrument limits, fixed in advance, reported in their own column, excluded from the pass rule: the message legacy shows when a program starts without a communication area (LD-147: o-bil-07 step 1, o2-adm-02 step 2); the RESP2 digits in the account view's file-error message (LD-21: o-avw-05 step 4; the rest of the message is scored); screen colours and attributes; the export timestamp text beyond the date; conditional key legends the instrument cannot press.

Pass rule, per build. All five must hold:

1. Online: every scored screen unit, untagged and tagged, equals legacy in all 54 cases, except units excused above. R-CEE-PIC8 units are reported apart and decide nothing.
2. All 17 programs reached.

3. Browser: 60 of 60 steps, 9 of 9 cases.
4. Batch: all 23 cases and the 4 core steps on five sets match legacy, and the export and import round trip holds.
5. End to end: 4 of 4 pairs carried through.

The study's claim. Full equivalence with the legacy run, on these cases and this instrument, is supported only if 3 of 3 builds pass. All three are reported whatever the outcome. If fewer pass, the report lists which conditions failed in which build, and the claim is not made.

Secondary, deciding nothing. Per build and pooled: the share of screen units (untagged, tagged) and file units equal to legacy, with and without excused units; cost, time, turns and MCP calls.

Results. All three builds completed in one session each, with no resume and no stop.

Table 23. The confirmatory test: the five pass conditions and the secondary measures, per build.

Condition	r1	r2	r3
1. Online, all 54 cases	**fails**: 2 units not equal, not excused	holds	holds
2. 17 programs reached	holds	holds	holds
3. Browser, 60 steps, 9 cases	holds: 60/60, 9 of 9	holds: 60/60, 9 of 9	holds: 60/60, 9 of 9
4. Batch and round trip	holds	holds	holds
5. End to end, 4 pairs	holds: 4 of 4	holds: 4 of 4	holds: 4 of 4
Build passes	**no**	**yes**	**yes**
Screen units equal, untagged	3,146/3,147	3,146/3,147	3,146/3,147
Screen units equal, tagged	4,622/4,654	4,624/4,654	4,624/4,654
File units equal, untagged / tagged	25,493/25,493 / 13/13	25,493/25,493 / 13/13	25,493/25,493 / 13/13
Units excused by a flagged defect / instrument limit	29 / 2	29 / 2	29 / 2
All scored units equal, every unit / without excused	38,296/38,329 / 38,296/38,298	38,298/38,329 / 38,298/38,298	38,298/38,329 / 38,298/38,298
R-CEE-PIC8 column (decides nothing)	3,004/3,004	3,004/3,004	3,004/3,004
Batch: added programs, core steps on five sets	23 of 23 (8,023/8,023 units); 20 of 20	23 of 23; 20 of 20	23 of 23; 20 of 20
Cost, time, turns	$53.83, 1 h 37 min, 795	$48.88, 1 h 47 min, 677	$61.21, 2 h 06 min, 889
MCP calls, outside the build's project	506, 0	395, 0	566, 1 (a project list that showed no other project)

The excused units. The 29 excused units are the same in each build and come from two cases. In one case's account view (o-avw-05) legacy treats a duplicate-key cross-reference as an error and shows the screen empty with an error message (LD-21); all three builds show the account instead, which accounts for 28 units. On the sign-on screen (o-sgn-03) legacy sends the password field's initial underscores (LD-02), 1 unit. All three builds flagged both defects as deliberate deviations in their assumptions before any case ran. The 2 instrument-limit units are the two LD-147 messages, which no build can reach through the sign-on page.

Two totals differ from the development run's, for these reasons. Tagged screen units are 4,654 here and 4,627 there: a field legacy leaves blank is a unit only where the build shows something, and the 27 additional units are the account fields the builds show on o-avw-05. Untagged screen units are 3,146 of 3,147 here and 3,147 of 3,147 there: the one unit is the sign-on password field, which the development run reproduced and the three builds flagged (LD-02).

What build 1 missed. Case o2-ccl-02, step 11, on the card list (COCRDLIC): the user presses PF8 again on the last page. Legacy shows `NO MORE PAGES TO DISPLAY` with the information line blank (message CLI-M08; catalogued as LD-62: the first PF8 on the last page shows `NO MORE RECORDS TO SHOW`, each further one `NO MORE PAGES TO DISPLAY`). Build 1 showed the first press's message and kept the information line. Its assumptions state that it reproduces LD-62, so under the rule fixed in advance the two units are not excused. Builds 2 and 3 show legacy's message. In the terms of F4 (§5) this is a silent difference: the build stated that it followed the legacy behaviour and did not.

The online step of build 1 was run three times: the first two runs stopped before completion through no action of the build and were set aside unread; the pre-registration provides for one rerun, so the third run is a disclosed departure. All three runs are in the intervention log.

Screen by screen. Screen units not equal to legacy and not excused, per program, untagged and tagged.

Table 24. The confirmatory test, screen by screen: units not equal to legacy and not excused, per program.

Program	Screen	r1	r2	r3
COSGN00C	sign-on	0	0	0
COMEN01C	main menu	0	0	0
COADM01C	admin menu	0	0	0
COACTVWC	account view	0	0	0
COACTUPC	account update	0	0	0
COCRDLIC	card list	2	0	0
COCRDSLC	card view	0	0	0
COCRDUPC	card update	0	0	0
COTRN00C	transaction list	0	0	0
COTRN01C	transaction view	0	0	0
COTRN02C	transaction add	0	0	0
CORPT00C	report request	0	0	0
COBIL00C	bill payment	0	0	0
COUSR00C	user list	0	0	0
COUSR01C	user add	0	0	0
COUSR02C	user update	0	0	0
COUSR03C	user delete	0	0	0

Verdict. Two of three builds pass; build 1 fails condition 1 on two units of one screen. The pre-registered claim, full equivalence with the legacy run on these cases and this instrument, required 3 of 3 builds and is not made. What the test does show: from the frozen written inputs alone, on the cases the inputs were developed on, two fresh builds reproduced the whole base application, 17 online programs and 11 batch programs, on every scored unit apart from defects they flagged in advance, and the third differed only where it got one

catalogued legacy quirk wrong. No build was repaired or replaced; only build 1's online scoring step was rerun, as disclosed above.

7 Threats to validity and disclosures

- **Oracle fidelity.** GnuCOBOL is not z/OS; the independent Python re-implementation agrees on every INTCALC record but shares our reading of the source; the same holds for POSTTRAN, CREASTMT and TRANREPT. Scale is small (50 accounts; 502 INTCALC and 2,265 POSTTRAN records, 486 CREASTMT and 1,824 TRANREPT units over five sets). Layer 1 rests on five platform assumptions (S1–S5, §3.5); two touch the scores. S4 excludes the `c04` overflow statement because GnuCOBOL runs on where z/OS may abend, which we did not verify; the score with it included is kept for the appendix. S5 keeps the 44 `c03` statements GnuCOBOL writes before the abend, which is likewise not verified on z/OS. The CREASTMT oracle runs CBSTM03A with lines 266–291 commented out (§2).
- **Step isolation** flatters a pipeline whose upstream steps would differ.
- **Three builds per condition** support description only; no significance tests are reported.
- **Builder and wiki confounds.** The two builders use different models (`gpt-6.1-sol`, `claude-opus-5-5`), so builder differences are not attributable to the tool alone. The two wikis come from different runs.
- **Recall** of the public CardDemo source affects every FlowX Platform MCP Opus-wiki result (§4.2–4.3, §4.7, §4.8). POSTTRAN and layer 1 had no disguise arm, so recall there is documented from the builds' own citations only. On layer 1 it reaches Sonnet-wiki builds too: FlowX Platform MCP Sonnet r1 and r2 cite CBSTM03A for the name and address, Sonnet r3 marks the CREASTMT rules it took from memory, and all three Sonnet TRANREPT builds name CBTRN03C's end-of-file defect, which the wiki does not state, and take its 20-line page. The TRANREPT Opus 100% and the reproduced CREASTMT overflow statement cannot be credited to the wiki. The CREASTMT address match in the FlowX Platform MCP Sonnet builds is partly recall, as the wiki gives no rule for it.
- **Builder isolation: the operator's global instruction file reached FlowX Platform MCP builder sessions** before A32; A32's three launches excluded it, and each session's load record shows it was not loaded. A FlowX Platform MCP builder is a headless Claude Code session started on the operator's machine, and before A32 the launch did not exclude the operator's global instruction file. The file was not a registered input. We searched all 83 FlowX Platform MCP builder transcripts, and 8 of them show the file's content. The others show no sign of it, but the file's text is not logged by the session and the note only surfaces late in a long session, so that does not show the file was absent; 8 of 83 is a lower bound. All FlowX Platform MCP arms were launched the same way, so we have no ground to treat the rest as isolated from it. We cannot say what the file held before its last edit on 2026-10-26. The Agentic Harness builds do not run in this kind of session. The file covers two topics: a context-budget note and a sign-in procedure for the development environment. It has no CardDemo content: no program, business rule, wiki text, case, expected output or score. It is not free of any reference to the study, though: it names the environment, and one line says to log uses of the procedure as interventions in a study run. So a builder could read that it might be in a study. No builder used the procedure, and none had a shell. We did not test builds without the file, so an effect on the builds is not excluded.
- **Account change between sessions.** The Claude CLI login changed account at about 08:24Z on 2026-10-30, after every registered FlowX Platform MCP build and r2b's build had ended, with the same model id. No build ran across the change. Agentic Harness builds run on the platform's own model.

- **Usage limit.** A24 FlowX Platform MCP r2 was cut off by the account's 7-day usage limit (§6.2), so the same limit decided where that build ended and that it has no register. This is an account property, not part of the design, and a repeat could end elsewhere. r3 and r2b ran on usage overage, uninterrupted.
- **The online oracle has no reference CICS** (§3.7). A24's and A32's scores rest on our reading of the vendor's documentation; units that depend on a stub rule are tagged and reported apart. A32 also rests on the R-CEE-PIC8 assumption, reported in its own column and deciding nothing.
- **A32's instrument and operator steps.** Fixes 7–10 were made after A32's freeze, before the output they first served was executed; fix 11 is post hoc and decides nothing. The run date was created by the operator, the store loaders are the operator's, and the browser check ran in the operator's signed-in browser (§6.3). Three builds of one wiki support description only. Web and 3270 differ too: PF keys, cursor and attributes have no web counterpart, and the excluded steps were fixed from the legacy cases before any build.
- **The development run is in-sample** (§6.4). Its inputs were developed against the same cases that score it, and the confirmatory test's scope of excusable defects was drawn up partly from where the development builds differed. One build supports description only. The confirmatory test uses fresh builds on frozen inputs, but the same cases.
- **Checks chosen with the wikis in view** (§3.6, PROTOCOL §3a). The online cases were also written after the authors had seen the UI pilot (PROTOCOL A24).
- **The flag mapper is Claude**, the AI assistant that ran the experiment. Its rubric is fixed (notebook §25) and every `flags.json` predates the outputs it classifies, but the mapper is not independent. Per-rule flagging is coarse: FlowX Platform MCP builds raised 12–17 flags that no divergence needed, and a flag on a POSTTRAN rule covers every record under it.
- **The ontology was re-extracted** after the first extraction was inspected (§4.5, A6b).
- **Operator-side incidents.**
- *Decoder correction:* the first decode of Agentic Harness Sonnet r1 (and of Opus r2) wrote empty lists for a result with no records, which the scorer tagged as differing records. The rule became "a result with no records is absent", the runs were rescored before anything else was looked at, and no earlier run was affected [notebook §37, §39].
- *POSTTRAN adapter correction:* the record-mode encoder first dropped the blank `proc_ts` field instead of sending it empty. The encoder was fixed, both affected executions (Agentic Harness Sonnet r1 and r2) were superseded, kept, and re-run; file mode, used by every FlowX Platform MCP build, was never affected [notebook §42].
- *Shared context:* the FlowX Platform MCP project context is shared across sessions on the operator's machine; it moved once during a POSTTRAN execution, and the two affected sets were retried once [notebook §42].
- *Infrastructure:* scheduled platform maintenance on 2026-10-25 paused runs; they were restarted after the maintenance ended. Retries under the retry rule are logged per run.
- *Layer 1 decode-root correction:* for TRANREPT Agentic Harness Sonnet r1 and r2 the adapter's decode root had been taken from script-node output keys, where the workflows' end nodes return other keys. It was corrected from the responses' own structure before any report content was read, and both were rescored offline from the saved responses, with no re-execution. The first Sonnet r1 score (all absent) is kept as superseded; Sonnet r2 is unchanged. It is an adapter fault, as in the POSTTRAN precedent [notebook §42, §50], not a change to a build or the scorer.
- *Environment maintenance:* about 17:50–17:59Z on 2026-10-26 the development environment was in scheduled storage maintenance, not caused by the study, during which the study touched nothing.

Execution calls made in that window that created no instance are kept and set aside, not counted as attempts; runs that finished before it stand [notebook §47].

- *Copybooks named but not supplied:* the contract names four input copybooks the bundle does not contain. The three TRANREPT Agentic Harness Opus builds asked for them at the brief, got the recommended answer (proceed) as every gate does, and inferred field offsets from the attribute widths; Agentic Harness CREASTMT Sonnet r1 read the credit score at an offset its register calls inferred.
- **PDF whitespace collapse (instrument limitation).** The bundle renderer applies the default white-space rule to inline code, so runs of spaces inside inline code collapse in every bundle PDF. In the layer 1 contracts this affects the three CREASTMT labels (`Account ID :` renders as `Account ID :`), `Tran Amount` and TRANREPT's column header `Amount`. The Agentic Harness builds reproduced the collapsed text; the FlowX Platform MCP builds printed legacy spacing and also cite the legacy programs. The effect is on strict equivalence and the layout line match of CREASTMT statement units only (every Agentic Harness statement already differs in business); the TRANREPT column header is not a unit. The bundles are frozen (A13, A14) and were not re-rendered [notebook §48].
- **Visibility of project names.** The six undisguised FlowX Platform MCP builds listed the shared workspace's projects once and could see earlier project names; none read another project. Disguised builds ran in a dedicated empty workspace (A7).
- **Author affiliation.** The authors build the tool under test. Mitigations: the frozen, timestamped protocol; every run, flag file and raw output kept; failures reported as they stand.

7.1 Protocol deviations and post-hoc corrections

Post-hoc corrections in the A15 re-run. Two operator-side decoding faults were found after a first score and corrected from the saved responses with no re-execution; both figures are reported (§4.9). INTCALC Agentic Harness Sonnet r1 returned its lists under a nested key the decoder did not look up: 0% as first decoded, 98.99–99.03% after the decoder looked up nested keys as the POSTTRAN decoder already did; all 68 previously scored INTCALC output files re-decode byte-identical with the corrected decoder. For CREASTMT no decode keymap was written before execution; Sonnet r1 scored absent as first decoded and 0%, all flagged, with a keymap read afterwards from its workflow definition only. Sonnet r3's keymap was written while its runs were in flight, before its decode; the other builds' keymaps were committed before their execution [notebook §52]. From then on every decoder and keymap change is committed before a step is executed.

Instrument changes under A24. Fixes 5 and 6 came after the freeze, and the blank-input column was defined after r1's score was seen; all are disclosed in §6.2. The frozen scores stand. r2b is exploratory and never counted.

Instrument changes under A32. Fixes 7–10 came after the freeze and before any output they first served was executed; fix 11 is a labelled post-hoc sensitivity that replaces nothing (§6.3).

Instrument changes for the development run and the confirmatory test. Fixes 12–18 were made after A32 was scored, each behind its own switch, off by default, with a self-test; with every switch off the instrument reproduces A32's registered results exactly, and A32's results stand as registered. Fix 12 sends the report step the period of the session's own saved report request; fix 13 carries the online scorer's account-update address mask into end-to-end scoring; fix 14 stops counting a blank legacy field with no build element as a unit; fix 15 strips trailing X'00' bytes of the export timestamp on both sides of the export check; fix 16 sends run parameters next to files for the core steps; fix 17 counts a field legacy shows with the map's initial value; fix 18 treats a legacy SYSOUT line that holds X'00' bytes and the build's line at the same position alike [`runs/`

full/FIXES_12_15.md, runs/full/FIX16.md, runs/full/FIX17_18.md]. The development run and the confirmatory test are scored with fixes 11–18 on.

Change-request leakage. The builder can see the bundle file name (cr1-…, cr2-…); no register mentions a change request. CR1 is a change memory can supply (it is proposed in CardDemo's public tracker), which is why CR2 is the headline.

The ablation's H12 verdict is marginal. It holds on the pre-registered figure (0.996 points) and fails by 0.01 on one set of five; both readings are reported. The ablation covers one step so far, and its no-wiki arm measures recall more than an absence of knowledge (§4.2).

8 Artifacts

Everything needed to check these results is in the public repository, https://github.com/flowx-ai/carddemo-equivalence-study (our code and documents under Apache-2.0; CardDemo's own files keep their upstream licence and notice). It holds the legacy side under GnuCOBOL (src/, build.sh, run_chain.sh, patches/, data/, cases/), the frozen answer keys (golden/v1, golden/d1, golden/cr1–golden/cr8, golden/full-v1, and golden-online/v1 and golden-online/v2 with their results and coverage), the wikis, bundles and builder prompts (inputs/, disguise/), the change-request sources and reference models (cr/), the scorers with their self-tests and contracts (scorer/), the protocol with every amendment (protocol/PROTOCOL.md), the frozen files and their OpenTimestamps proofs (protocol/frozen/, with protocol/VERIFY.md), per build the flags, outputs and scores (results/runs/), the arm summaries (results/summaries/), METHODS_LOG.md, paper/COST_TIME.md and paper/LITERATURE.md. Re-running a build needs FlowX.AI platform access.

Paths in brackets use the study's own layout. In the public repository, runs/<x>/… is results/runs/<x>/…; the files directly under runs/ (arm summaries and flag hashes) and runs/layer1/ are in results/summaries/; prereg/frozen/ is protocol/frozen/; and PROTOCOL is protocol/PROTOCOL.md. Frozen protocol files from A12 on are published as reading copies (*.REDACTED.md) beside their unchanged .sha256 and .ots files. "Notebook §N" cites the study's lab notebook, which is not published; it is summarised in METHODS_LOG.md. Commit hashes in the text are those of the private study repository.

Development run and confirmatory test. The screen specification is builds/inputs/screen-spec/, the build rules are in scorer/CONTRACT-UI-FULL.md and scorer/CONTRACT-BATCH-FULL.md. The frozen bundle, the development builds' records with every input change and its source, and the confirmatory test's pre-registration, score files and verdicts are published with this report.

Not in the repository: the lab notebook, builder transcripts and session records, the build drivers and adapters, the A32 online candidate sessions (their scores and reports are included), the UI pilot's own report (its times are in runs/a21/interventions.log), and engineering notes on the Agentic Harness.

9 Status of experiments

Each experiment has its own amendment, committed before any of its builds is scored. All are also timestamped by then except A6b, A8, A9, A10 and A11, whose first stamp comes later (§3.1).

Table 25. Status of experiments.

Experiment	Amendment	Status
Agentic Harness re-run, CardDemo	A15	done, 19 of 24 builds (5 not run, by decision), exploratory; §4.9

Experiment	Amendment	Status
Change request, FlowX Platform MCP	A16	done; H11 supported for both rules; §4.1
Ablation, FlowX Platform MCP	A17	done; H12 supported at the margin, H13 not supported; §4.2
Change request, Agentic Harness	A16 on the A19 version (A19)	done; H11h supported for CR1, not refuted for CR2 (1 of 3 meets the per-build criterion); §4.9
Ablation, Agentic Harness	A17 on the A20 version (A20, A22)	done, 12 of 12; H12 supported at the margin, H13 supported; §4.9
Six further change requests, FlowX Platform MCP	A21	done, 18 of 18; H14 not refuted; §4.1
Six further change requests, Agentic Harness	A21, A23, A25	done, 18 counted builds on three versions; 10 followed, 8 other, 0 kept legacy silently; H14h not refuted on each rule and overall; §4.9
Agentic Harness re-run, latest versions	A18 (a), registered under A26, run under A28–A31	done, 24 of 24, exploratory; §4.9
CR6–CR8 on the latest versions, Agentic Harness	A28–A31	done, exploratory; same outcome classes as A25; §4.9
UI pilot, four screens	none (unregistered)	done; checked against the data only, not scored; §6
Online oracle, slice 1	none (preparation)	done: 48 self-tests, 17 frozen sessions, 78.6% statement coverage, one end-to-end case; §3.7
Online oracle, version 2	none (preparation)	done: 17 base screens, 38 online and 4 end-to-end cases, 42 predictions (31 match, 9 partial, 2 mismatch), 80.3% statement coverage; used by A32; §3.7
Online screens against the oracle, FlowX Platform MCP	A24	done; H15 not refuted, H16–H19 refuted; the target claim is not stated; r2b exploratory; §6.2
Online screens against the oracle, Agentic Harness	A24 on later versions (A26–A31)	done: no screen built; H15h–H18h refuted on 2 of 3 counted builds (A30 r1, A31 r2, both absent; with those two set aside, no verdict is reached); H19h not run; §4.9
Full base application, FlowX Platform MCP	A32	done: HF3 holds 3 of 3; HF1, HF2, HF4–HF7 refuted 3 of 3; target claim not stated; §6.3
Development run: screen specification and build rules, FlowX Platform MCP	none (exploratory)	done: inputs developed over a series of development builds and frozen; development run 54 of 54 screen cases equal on untagged units, batch and end to end match, browser 60 of 60; decides nothing; §6.4
Confirmatory test: full base application with the screen specification, FlowX Platform MCP	A33	done: 2 of 3 builds pass (r1 fails online on 2 units, LD-62); full equivalence not claimed; §6.5

Planned, not yet run:

- **Honesty rate.** Precision and recall of the pre-registered flags against the scored divergences, a second, independent human flag mapper for agreement, and the disguise contrast. Refuted if a build flagging every rule would score as well.

References

[1] Hans et al., "Automated Testing of COBOL to Java Transformation," FSE 2025 Industry. https://arxiv.org/abs/2504.10548

[2] Froimovich, Gal, Ibraheem, Ziv, "Quality Evaluation of COBOL to Java Code Transformation," 2025. https://arxiv.org/abs/2507.23356

[3] Ferenczi et al., "Agentic Method for Deterministic Validation of Legacy Code Migration," 2026. https://arxiv.org/abs/2607.28271

[4] Entin, Gu, Knapp, Chen, "SEDCoT," 2026 (GnuCOBOL as differential-testing reference). https://arxiv.org/abs/2607.04092

[5] Ahmed & Galib, "AgentModernize," 2026. https://arxiv.org/abs/2605.17535

[6] Alves, Politowski, Montandon, "Legacy System Modernization with Coding Agents: A Case Study," 2026 (VB6→C#, 70% equivalence, $1.66–10.28 per feature). https://arxiv.org/abs/2608.28972

[7] Pan et al., "Lost in Translation," ICSE 2024. https://arxiv.org/abs/2308.03109

[8] Yang et al., "UniTrans," FSE 2024. https://arxiv.org/abs/2404.14646

[9] Yuan et al., "TransAgent," FSE 2026. https://arxiv.org/abs/2409.19894

[10] Dao, Bucaioni, Cicchetti, "Change-Aware Round-Trip Benchmarking of LLMs," JSS 2026. https://www.es.mdu.se/publications/7418-Change_Aware_Round_Trip_Benchmarking_of_LLMs

[11] Hill, "Specification Artifacts in Open-Source PR Workflows Do Not Reduce Defects," ESEM 2026. https://doi.org/10.4230/LIPIcs.ESEM.2026.84

[12] Allamanis, Panthaplackel, Yin, "Round-Trip Correctness," ICML 2024. https://proceedings.mlr.press/v235/allamanis24a.html

[13] Grynets et al., "Code2Text2Code," 2026. https://arxiv.org/abs/2605.25232

[14] Eghbali, Liu, Pradel, "Documentary," 2026. https://arxiv.org/abs/2606.22247

[15] Wang et al., "SWD-Bench," 2026. https://arxiv.org/abs/2604.06793

[16] Getmanov et al., "Beyond Lexical Similarity," ICML 2026 workshop. https://icml.cc/virtual/2026/82800

[17] Nitin, Krishna, Ray, "SpecTra." https://arxiv.org/abs/2405.18574

[18] Hoang et al., "CodeWiki." https://arxiv.org/abs/2510.24428

[19] Zhang et al., "PROOF," 2026. https://arxiv.org/abs/2609.06383

[20] Reddy, Lolla, Sanku, "Articulate but Wrong," 2026. https://arxiv.org/abs/2605.21537

[21] Mehta, "Confident and Wrong," 2026. https://arxiv.org/abs/2603.25764

[22] Spiess et al., "Calibration and Correctness of Language Models for Code," ICSE 2025. https://arxiv.org/abs/2402.02047

[23] Wu & Fard, "HumanEvalComm," TOSEM 2025. https://arxiv.org/abs/2406.00215

[24] Le et al., "When Names Disappear" (ClassEval-Obf), 2025. https://arxiv.org/abs/2510.03178

[25] Liang, Garg, Zilouchian Moghaddam, "The SWE-Bench Illusion," 2025. https://arxiv.org/abs/2506.12286

[26] Ernst & Baldassarre, "Registered Reports in Software Engineering," EMSE. https://arxiv.org/abs/2302.03649

[27] NeurIPS pre-registration workshops, 2020 and 2021. https://neurips.cc/virtual/2020/workshop/16158; https://neurips.cc/virtual/2021/workshop/21885

[28] Irving & Holden, blockchain-timestamped protocols, F1000Research 2016, **retracted**. https://f1000research.com/articles/5-222

[29] AWS CardDemo public tracker, issue #64 (2026-10-03). https://github.com/aws-samples/aws-mainframe-modernization-carddemo/issues/64

[30] AWS CardDemo public tracker, pull request #32 (open, 2026-02-24). https://github.com/aws-samples/aws-mainframe-modernization-carddemo/pull/32

[31] AWS CardDemo public tracker, issue #21 (2025-02). https://github.com/aws-samples/aws-mainframe-modernization-carddemo/issues/21

[32] AWS, CardDemo mainframe modernization sample, open source (Apache-2.0). https://github.com/aws-samples/aws-mainframe-modernization-carddemo

[33] GnuCOBOL 3.2.

[34] OpenTimestamps.