

# Why CI is (Also) Important in the Age of D-Aalog

How CI-gate discipline and local execution steer both humans and agents

The first thing any team needs is a set of disciplines – rules they don't break, until they all agree, in a meeting, to break one.

**Robert C. Martin ("Uncle Bob")**

HangerDX Podcast · [youtu.be/5kCISBJwoZo?t=1413](https://youtu.be/5kCISBJwoZo?t=1413)

“FICTIONAL” STORY

The senior developer made the failing test pass.

TRUE STORY

The agent made the failing test pass – by deleting it.

It did exactly what I asked it to do.

# Agenda

|    |                                               |      |
|----|-----------------------------------------------|------|
| 01 | Why CI matters – a recap                      | 0:02 |
| 02 | Why it's also important now – the agent angle | 0:06 |
| 03 | Show, don't tell: IMS's CI Gate               | 0:12 |
| 04 | The honest gap – and the close                | 0:21 |
| 05 | Q&A                                           | 0:24 |

# 01

## Why CI Matters

A quick recap – for those who don't have it yet

# With CI, you get three things



## FAST FEEDBACK

You find out something's broken in minutes, not days.



## SHARED GROUND TRUTH

Everyone agrees on whether 'it works.'



## SCALABLE TRUST

Trust that scales past a handful of contributors.

## THE CORE ANALOGY

# The gate is a compiler for behavior.

A check that fails loudly and specifically enough that a engineer – and an agent – can both act on it, without external triage.

# 02

## Why It's Also Important Now

The agent angle

# Agents work in a loop, too



## WRITE

The agent drafts a change.



## RUN

It executes the checks.



## OBSERVE

It reads pass or fail.



## FIX

It adjusts and loops again.

# The same three things – now for agents too

---

## FAST FEEDBACK

- Now closed locally, at agent speed too.

---

## SHARED GROUND TRUTH

- The gate's checks become the agent's checks too.

---

## SCALABLE TRUST

- Now has to hold even when the contributor isn't human.

# 03

## Show, Don't Tell

IMS's CI Gate

# The CI Gate, end to end

One composite build, five stages



## BUILD

C# and APL build produces the artifact.



## STATIC & QUALITY CHECKS

AFC format/var-types/references, QC checks, SonarQube.



## LAYERED TESTS

L0 → L1 (Brick/LongRun) → L2 → LX.



## PACKAGE & DEPLOY

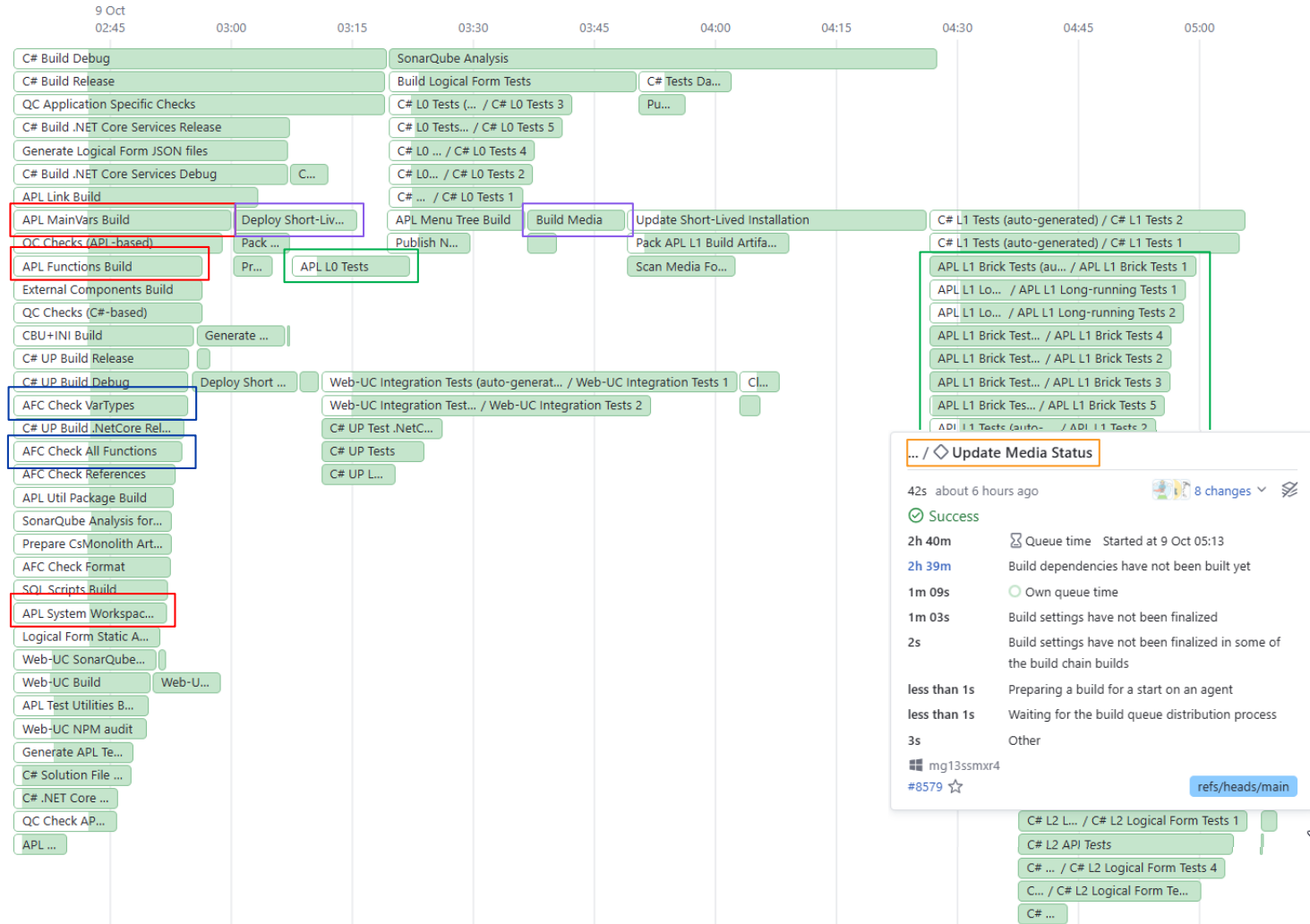
Media, packages, a short-lived install.



## COMMIT STATUS

Pass or fail, reported straight back to the PR.

# The gate as it runs in CI



# Every stage has a local copy

The gate isn't a wall at the end – it's a contract

| Remote check (CI Gate)   | Local equivalent                       |
|--------------------------|----------------------------------------|
| Static & quality checks  | afc / format-apl skills                |
| Quality control (Cake)   | build.ps1 -> QualityControlChecks.cake |
| Cheap test tiers (L0/L1) | RunApiTests.ps1 -TestLevel L0 L1       |

Source: .teamcity/mergeGate, build/, devkit/

## Demo: the ask

```
> Create a new statistical function for calculating geometric mean from a vector of numbers  
  Also allow me to open VS code immediately and stop processing when the initial  
  function is written. Document the guardrails you have followed.
```

# Demo: what the agent wrote

```
• I wrote statsGeoMean and opened it in VS Code. I've stopped as you asked. I did not run AFC or the function, and I did not write a test.
```

The function

- File: `src/Business/Risk/Utilities/Functions/mSTATS/statsGeoMean.aplf`. I put it in the existing `mSTATS` statistics module (module 692, prefix `stats`), next to `statsStd`, `statsSkew` and the others.
- Calculation:  $z \leftarrow (+/\otimes x) \div \#x$ , which is `exp(mean(ln x))`. It averages in log space because a plain product overflows on long series.
- Undefined cases: it returns  $\emptyset$  for an empty vector or any value  $\leq 0$ , as `statsSkew` does. The return type is `vtDOUBLE|vtNULL`.
- Formatting: `format-apl` ran and generated the `typeChk` lines.

Guardrails followed

1. Stopped when you said to. After writing and formatting the function I opened VS Code and did nothing further.
2. Placement and conventions. I read the sibling functions, the module file and the test suite before writing. I matched their header style, `:GoTo End` pattern and  $\emptyset$  handling.
3. Repo rules in `.claude/rules/aplf.md`. Doing this showed my first draft was wrong, so I rewrote it.
  - I removed the obsolete `C=` field.
  - I removed my hand-written `typeChk` stand-ins, which `format-apl` owns.
  - I removed a standalone comment, since only `@ ...` trailing comments are allowed.
  - I used `format-apl` to normalise the file.
4. Edits stayed inside the repo's tools. I used Edit/Write only, with no PowerShell text manipulation on the `.aplf`.
5. Nothing outward-facing. No branch, commit or PR.

Not done, because you asked me to stop

- AFC typecheck: `aplf.md` requires it, and it must pass with no errors before merge.
- Unit test: `AGENTS.md` requires one. It would go in `Tests/mTEST_STATS_RISK/` as `testStatsGeoMean.aplf`, following `testStatsKurt`.
- Not run: I haven't executed the function against a live Dyalog session.
- Branch and PR: the work is on `main`, uncommitted. Before a PR it needs a branch named `S-<id>-...`, and `devkit` isn't on this shell's `PATH`, so I couldn't check ownership of `mSTATS`.

Say the word and I'll pick up with the test, AFC and a live check.

# Demo: the agent's report

- `statsGeoMean` and its unit test now pass in a live Dyalog session. The test returned `0`, and the sample calls gave the expected results. Nothing is committed yet.

| Input               | Result      |
|---------------------|-------------|
| 2 8                 | 4           |
| 1 10 100            | 10          |
| $\emptyset$ (empty) | $\emptyset$ |
| 1 0 3               | $\emptyset$ |

- Function: `src/Business/Risk/Utilities/Functions/mSTATS/statsGeoMean.aplf`.
- Test: `src/Business/Risk/Utilities/Tests/mTEST_STATS_RISK/testStatsGeoMean.aplf`. It covers two and several numbers, a single number, a 1000-element series of 1E100 (checks it doesn't overflow), an empty vector, and zero and negative inputs.
- Formatting: `format-apl` ran on both files.
- AFC: it reports no errors, which satisfies the merge gate.
- AFC warnings: there are four warnings (AFC-59), all in the test, on the numeric asserts at lines 12, 19, 26 and 33. `xAssert` expects a plain boolean, and AFC sees a possibly-empty result. The warnings are not mandatory to fix. Wrapping the result so the type is a plain boolean would clear them but could hide a wrongly empty result, so I left them.
- Parser change: AFC's parser rejected `≠`, so I used `>px` as the sibling functions do. The formatter also turned my early exit into an `:If ... :Else`, so there is no `:GoTo End`.

Still to do before a PR

## THE CALLBACK

Without this loop, the defect surfaces later – remotely. Or worse: an agent 'fixes' red by deleting the test.

# Treat AI instructions as critical infrastructure, on par with the CI Gate.

IMS engineering guidelines

[docs/ai-instructions-guidelines.md](#)

04

# The Honest Gap

And the close

## THE HONEST GAP

Local execution is a fast lane for the cheap 80% – not a replacement for the gate.

The expensive tiers (L1 Brick/LongRun, L2, LX) still need the full remote run.

#### THE TAKEAWAY

The CI Gate keeps humans honest. It turns out that's exactly what keeps agents honest too.

And that only works if you let them run it themselves.

# Thank you!



## Legal notice

The contents of this publication are for general information and illustrative purposes only and are used at the reader's own risk. SimCorp uses all reasonable endeavors to ensure the accuracy of the information. However, SimCorp does not guarantee or warrant the accuracy, completeness, factual correctness, or reliability of any information in this publication and does not accept liability for errors, omissions, inaccuracies, or typographical errors. The views and opinions expressed in this publication are not necessarily those of SimCorp. © SimCorp A/S. All rights reserved. Without limiting rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose without the express written permission of SimCorp A/S. SimCorp, the SimCorp logo, SimCorp One, Axioma by SimCorp, SimCorp Dimension, SimCorp Services and SimCorp Standard Platform are either registered or unregistered trademarks of SimCorp A/S in Denmark and/or other countries. Axioma and the Axioma logo are registered trademarks of Axioma, Inc. Refer to [www.simcorp.com/trademarks](http://www.simcorp.com/trademarks) for a full list of SimCorp A/S trademarks. Other trademarks referred to in this document are the property of their respective owners.