Real World Testing Strategies for z/OS Application Development

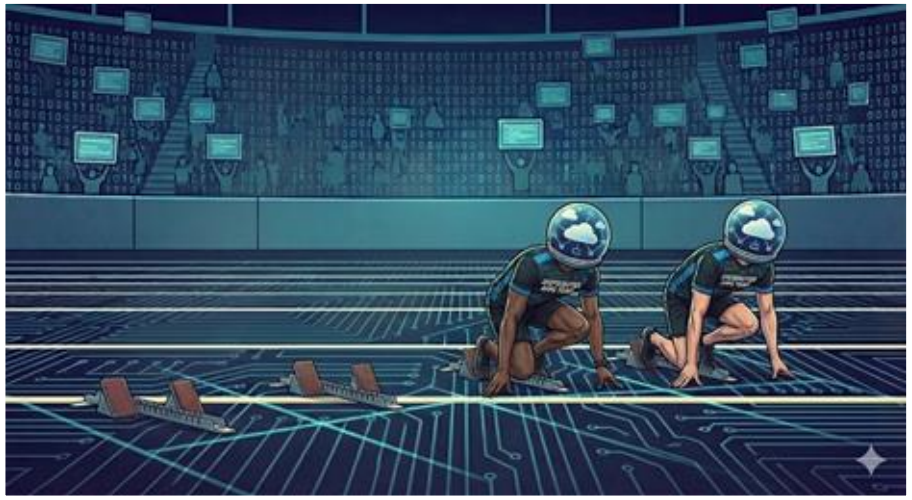
Chandru Rengarajan
Technical Consultant
chandru.rengarajan@broadcom.com

Petr Vacula
Product Manager
petr.vacula@broadcom.com

The State of Mainframe Testing

DEV	TEST	PROD
• OBSERVE • DISPLAY Mania • DEBUGGER **Challenge**: Speed of development impacted by application and environment complexity	• FUNCTIONAL • INTEGRATION • END-TO-END **Challenge**: Lengthy manual test cycles	• PERFORMANCE MONITORING • PROBLEM DIAGNOSIS **Challenge**: Disruptive and costly escaped defects

Business Push on Faster Delivery



SPRINT PREP

AUTOMATION PREP

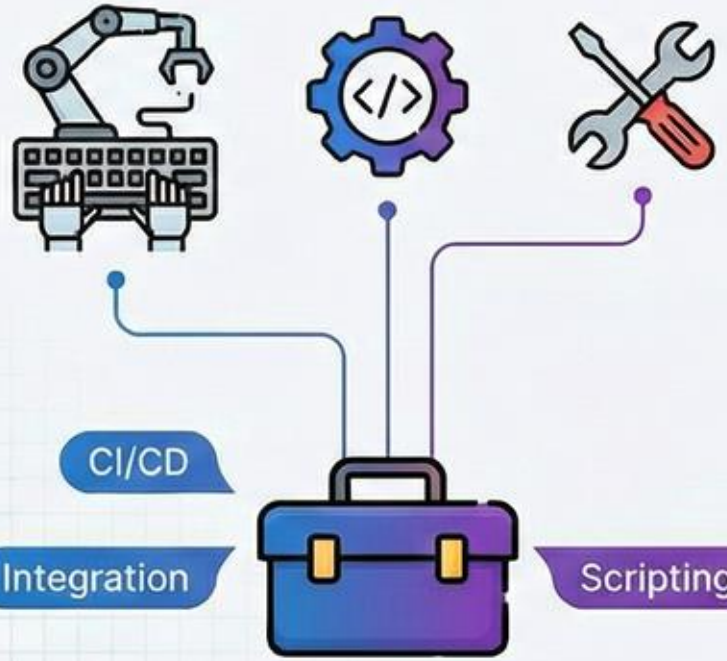
Define and Refine Your ~~GOLD~~ GOAL

Define modernization and improvements in outcomes (not tools)

HOW
to automate testing?
(Tooling)

Frameworks

Libraries



CI/CD

Integration

Scripting

Strategic Decisions

WHERE
to automate testing?
(Return on Investment)

High-Volume/Repetitive

Regression Suites

Critical Paths

Critical Paths

Stable Features

TEST STRATEGIES

From concrete plans to general approaches

Test Like the Distributed Folks Do (1/3)

SHARE
EDUCATE • NETWORK • INFLUENCE

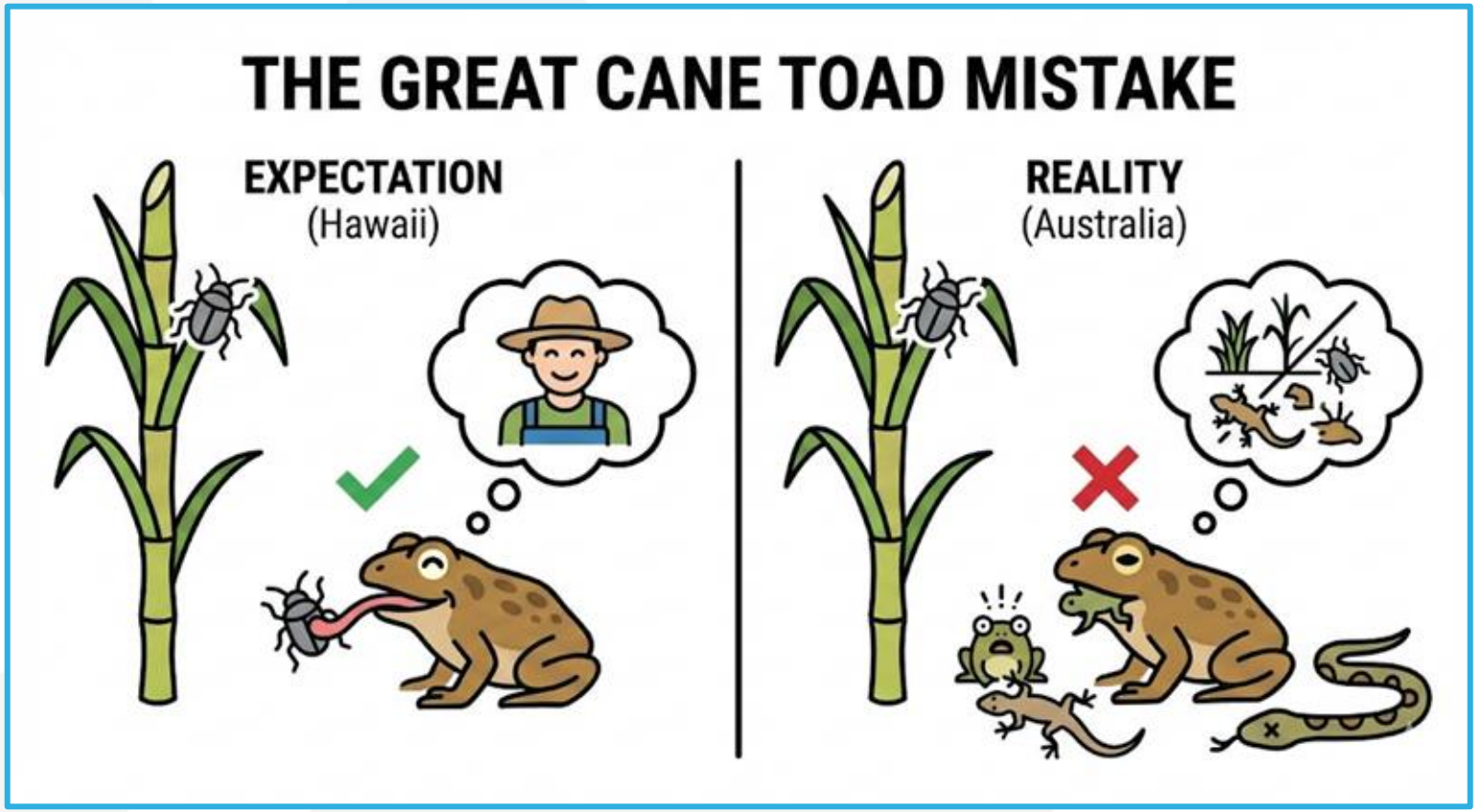
Goal: quality-first **without** copying wrong assumptions from distributed

Copy the **tooling patterns** (CI, reports, artifacts)

Don't copy the **platform assumptions** (stateless, cheap envs, easy mocks)

Focus areas today: **Do/Don't**

Contextual blindness



Test Like the Distributed Folks Do (2/3)

SHARE
EDUCATE • NETWORK • INFLUENCE

What to Adopt from Distributed Testing

- Standardize results: **JUnit XML** + publish test artifacts
- Run tests in CI: **small per change**, broader nightly/regression
- Treat everything as code: **JCL/PROCs/copybooks/config/test data defs**
- Build a layered strategy: **unit → component → integration → small E2E**
- Add quality gates: pass/fail + static analysis + regression thresholds

Test Like the Distributed Folks Do (3/3)

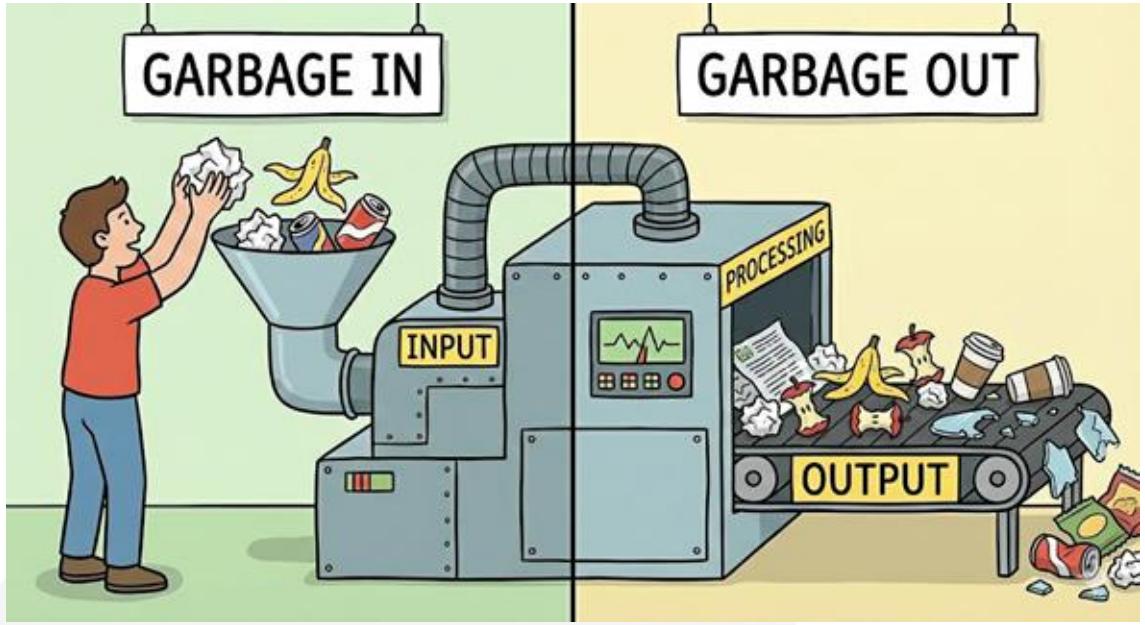
SHARE
EDUCATE • NETWORK • INFLUENCE

Common Failure Modes When Copying Distributed Patterns

- Don’t assume **stateless** apps or easy environment reset
- Don’t rely on **RC=0** as “testing” (assert business correctness)
- Don’t overdo brittle E2E (e.g., “drive everything via 3270”)
- Don’t mock everything—keep real **DB2/CICS/locking/encoding** coverage
- Don’t ignore shared-state flakiness: datasets, DB2, ENQs, schedulers

Test Data

- 2 fundamental approaches: generate or mask
- testing and test scenarios are only as good as the input test data
- consistency - data in shared LPARs
- scale - production like data



Test Data Management

SHARE
EDUCATE • NETWORK • INFLUENCE

Make Tests Repeatable

- Create **golden datasets** + version them (inputs + expected outputs)
- Control state: reset datasets/DB2 to **known baseline** every run
- Use isolation: **HLQ per team/build**; DB2 **schema/qualifier per run**
- Automate refresh: IEBCOPY/IDCAMS/REPRO + DB2 unload/load scripts
- Mask/obfuscate sensitive data; document field rules + edge cases
- Capture “data version ID” in test reports for reproducibility

Test Environments for z/OS

SHARE
EDUCATE • NETWORK • INFLUENCE

Stability + Isolation + Observability

- Define tiers: **Dev/Unit** → **Integration** → **Pre-prod/Regression**
- Keep runtime consistent: compiler/LE options, exits, PARMLIB, subsystem levels
- Isolate resources: HLQs, DB2 qualifiers, CICS regions where possible
- Prevent flakiness: manage **ENQs/locks**, shared queues, background jobs
- Capture run metadata: subsystem/region versions, configs, jobnames, traces
- Add observability: retain logs/dumps + performance baselines per suite

Better DevX

SHARE
EDUCATE • NETWORK • INFLUENCE

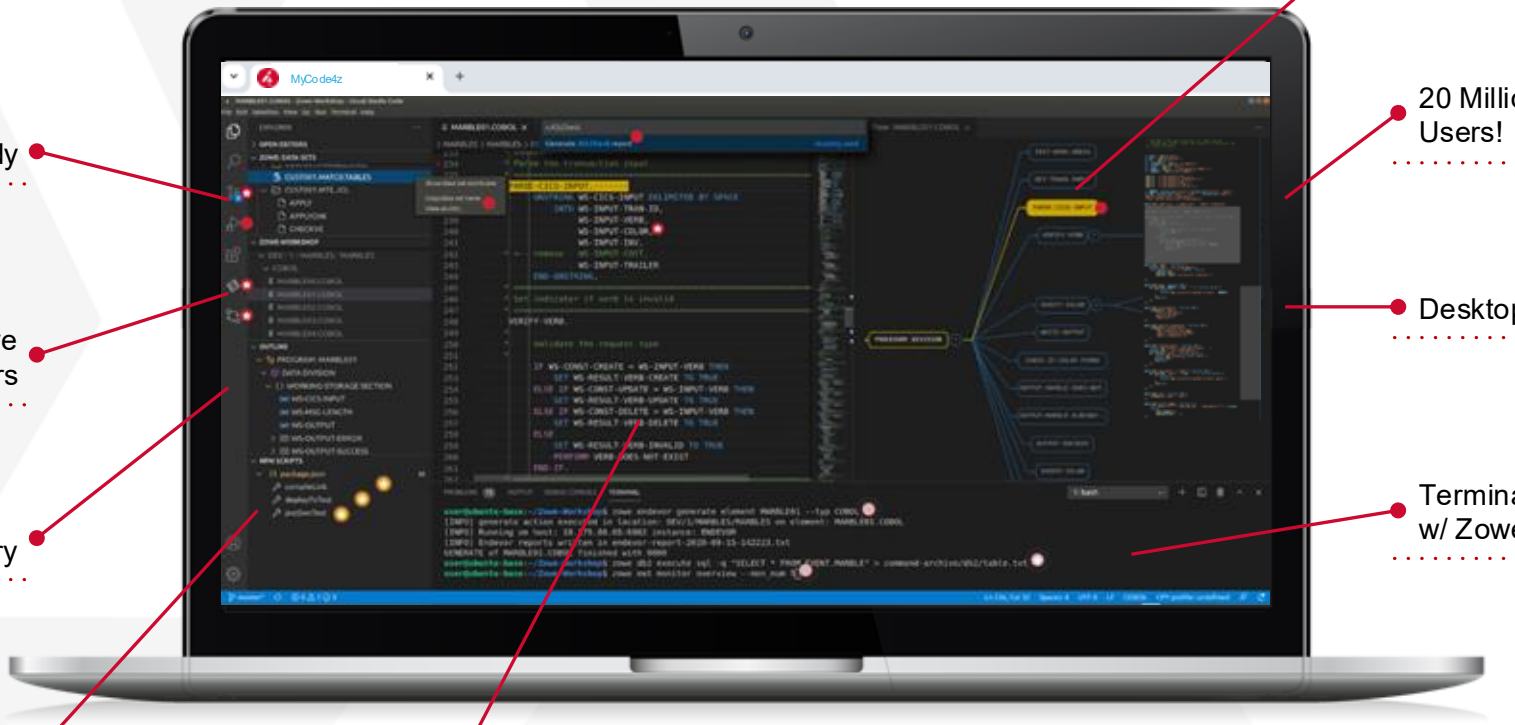
What?! Improved developer experience leads to higher quality?

Git Ready

Endevor & Zowe Explorers

Extension Library

Script Libraries (e.g., NPM)



Monolith Navigation

20 Million VS Code Users!

Desktop or Browser

Terminal Commands w/ Zowe CLI Plug-ins

IntelliSense, etc. w/ z languages

Eliminates slow paging; jump anywhere

code4z.broadcom.com

Info & Request a Workshop

Static Application Analysis

- Static scanning - quality & security
- Very usable and quick to benefit from

Scanning of application source to:

- Find bugs
- Enforce code styles
- Find security issues

Best Practices:

- Enforce quality gates

Tooling:

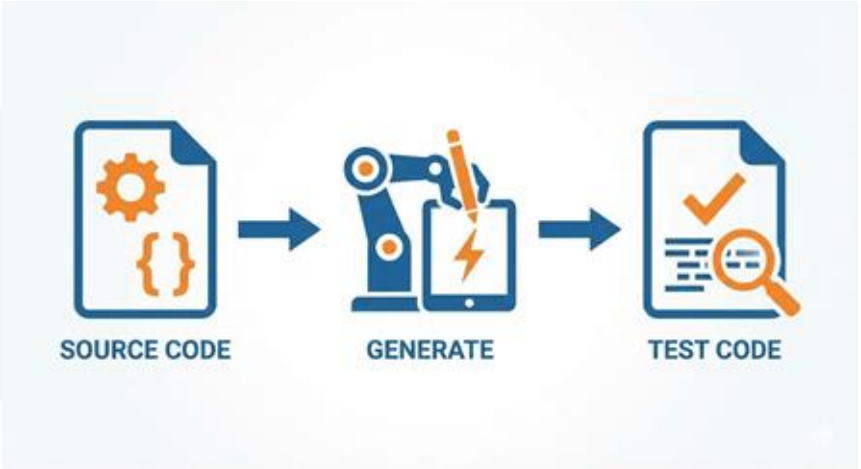
- SonarQube, Checkmarx, Veracode

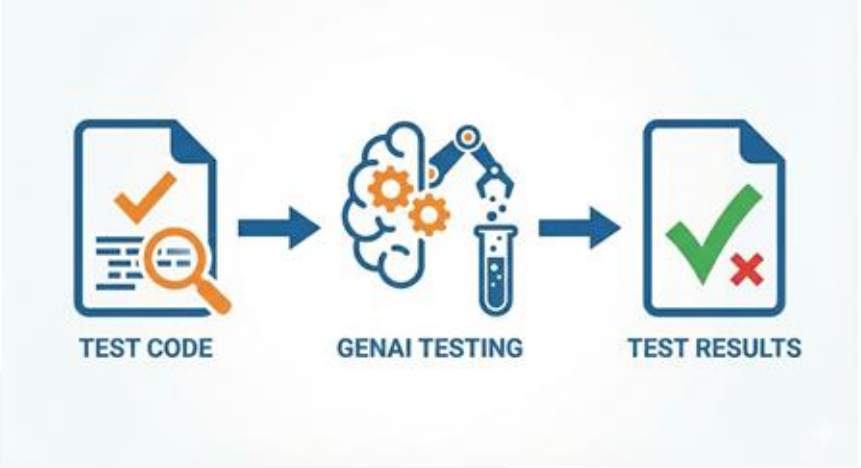
AI in mainframe DevOps

? where it helps safely

? create a test script vs. do the testing

? what happens when you run out of tokens

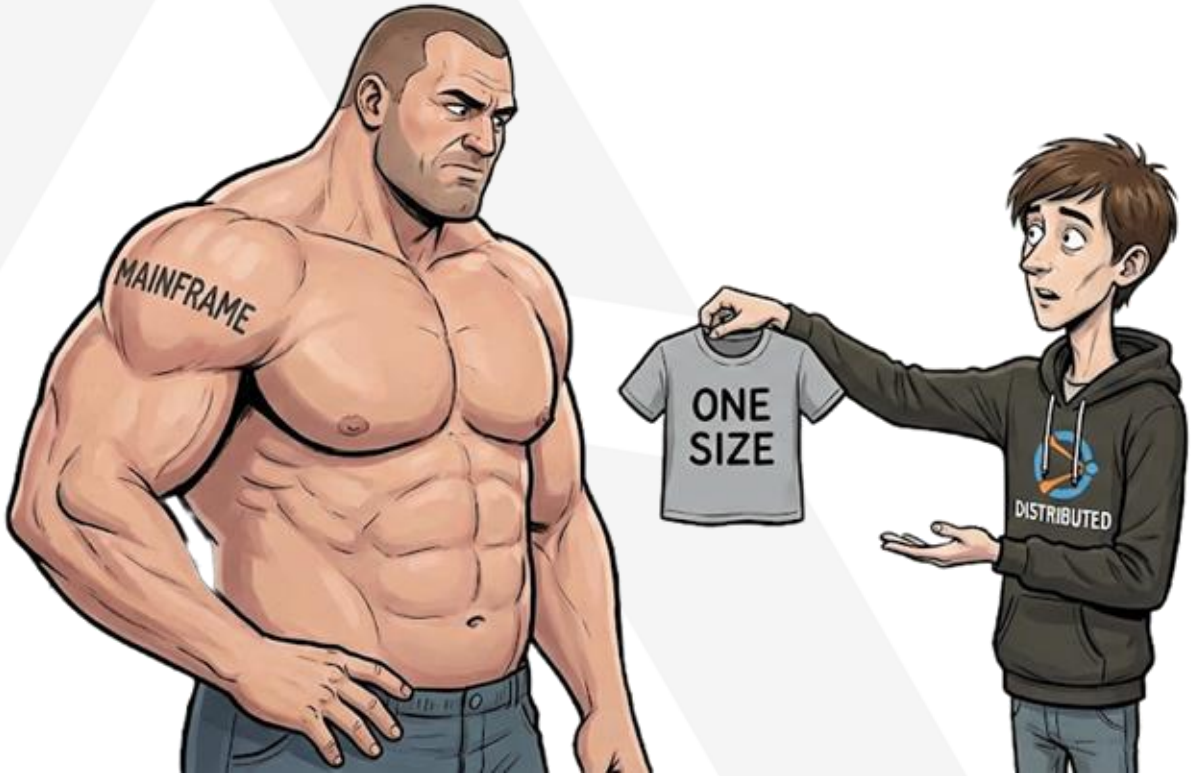




WHAT TESTING STRATEGIES DO YOU EMPLOY?

Takeaway

- Set your **goals** first
- **Stretch** before sprinting
- Consider together
  - **HOW** to do it **&**
  - **WHERE** to start
- Be **contextual** aware



Join a Mainframe Technical Exchange

Apr. 21-23, 2026	June 23-25, 2026	Oct. 20-22, 2026
European MTE Prague, Czech Republic	Virtual MTE Held Virtually	North American MTE Plano, TX

- Network with peers and Mainframe technical experts
- Participate in technical how-to sessions and hands-on workshops
- No registration fee!

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation