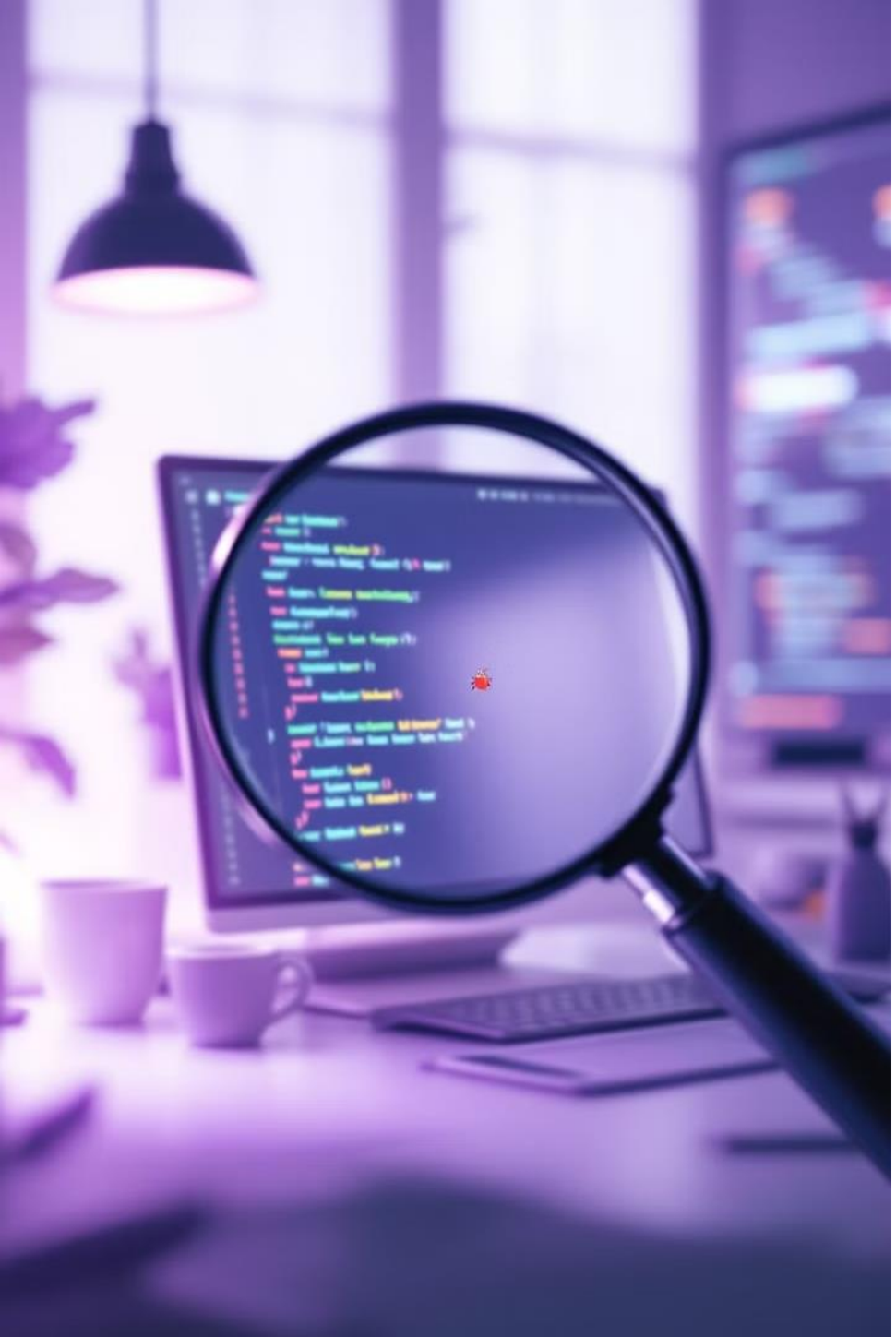


Testing Software Before Release: It's Just Common Sense

Ensuring Quality, Preventing Disaster

Presented by: Dr. Rich Greene, PhD

NYTUG

May 13, 2025

Ascert


VERSATEST
KILLS SOFTWARE
BUGS
STARSHIP TROOPERS

Common Sense Video

Common Sense in Software Testing

Testing isn't just good practice—it's essential protection.

Would you...

Drive a car without brakes being tested?

Cross a bridge with untested supports?

Fly in a plane with unchecked engines?

Then why...

Deploy code without thorough testing?

Risk your data with untested security?

Jeopardize user trust with buggy features?

Software risks are real

Financial losses can be devastating.

Reputation damage can be permanent.

User safety might be compromised.

Common Sense in Software Testing

Common sense in software testing means applying logical reasoning to ensure your product works as intended before users get their hands on it.

Verify It Works

Test basic functionality before release, just like you'd test drive a car.

Protect Your Reputation

Bugs damage trust and cost more to fix after release.

Consider User Experience

Put yourself in users' shoes and test accordingly.

Invest Time Upfront

Thorough testing saves time and resources in the long run.

Why Do We Test Software?

Software Is Everywhere

Banking, healthcare, transportation, and entertainment all rely on software.

Bugs Cost More Later

Finding bugs before release is significantly cheaper than after launch.

Preventing Disasters

A single bug can cause financial loss, security breaches, and even harm lives.

Real-World Software Failures

1

NASA's Mars Climate Orbiter (1999)

A unit conversion error caused a $125 million spacecraft to burn up in Mars' atmosphere.

2

Knight Capital Group (2012)

A trading software glitch led to a $440 million loss in just 45 minutes.

3

Therac-25 Radiation Machine (1985-1987)

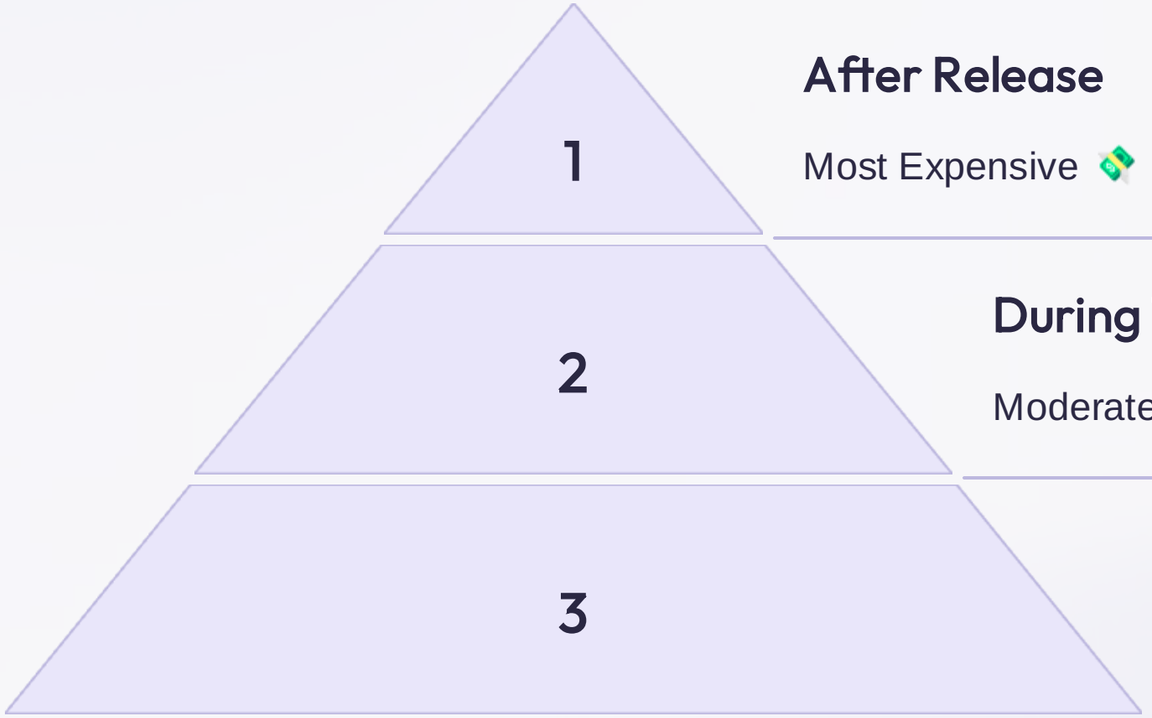
A software bug caused lethal radiation doses, resulting in at least 6 patient deaths.

4

Windows 10 Update (2018)

A poorly tested update deleted users' personal files permanently.

The Cost of Fixing Bugs



After Release

Most Expensive 💸

During Testing

Moderate Cost 💵

During Development

Least Expensive 💰

Fixing a bug after release can cost 100x more than catching it early.

Skipping testing isn't saving time or money—it's delaying disaster.

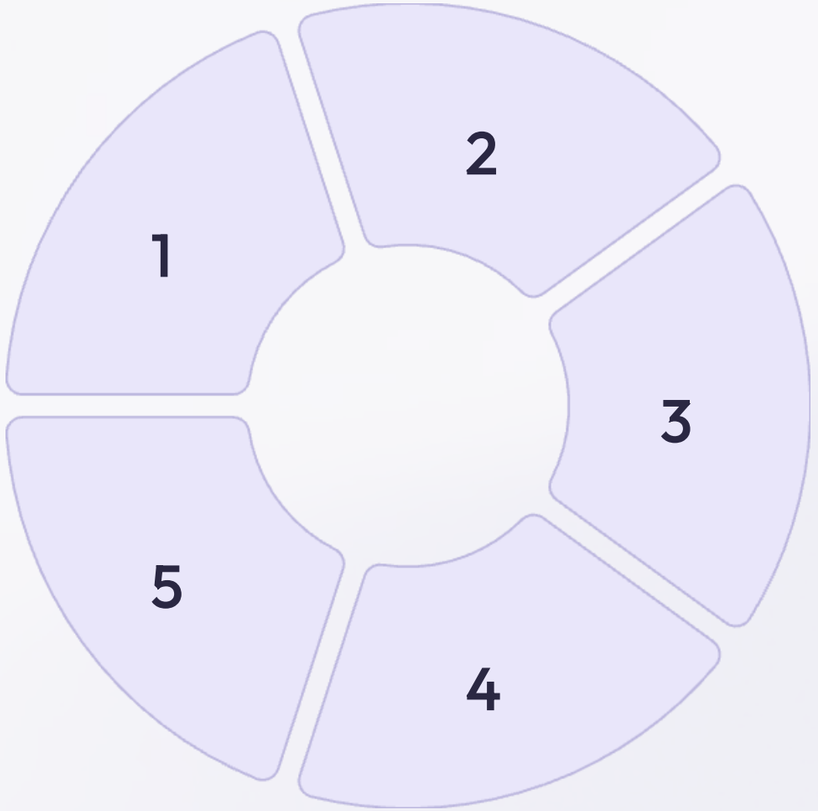
Types of Software Testing

Unit Testing

Tests individual components. Catches small bugs early.

User Acceptance Testing

Performed by real users. Ensures software meets needs.



Integration Testing

Ensures components work together. Prevents module miscommunication.

System Testing

Evaluates the entire application. Identifies high-level issues.

Performance & Security

Tests speed, stability, and security vulnerabilities.

Testing in Agile & DevOps

Continuous Testing

Testing happens throughout development, not just at the end.

Automation

Automated tests run every time code changes.

Collaboration

Developers and testers work together to prevent bugs.

Deployment

If it's not tested, it's not ready for release!

Modern development requires testing as a continuous process, not a final step.

Best Practices for Effective Testing

Start Testing Early

Begin in early development stages. The earlier a bug is found, the cheaper to fix.

Automate Strategically

Automate repetitive tests, but don't ignore manual testing for user experience.

Test Real-World Conditions

Test on different devices, networks, and environments to prevent unexpected failures.

Continue After Release

Post-release monitoring helps catch issues that slipped through testing.

What testing tool can do all of this?

VersaTest Automator

Your complete solution for modern testing challenges.

Automated Testing

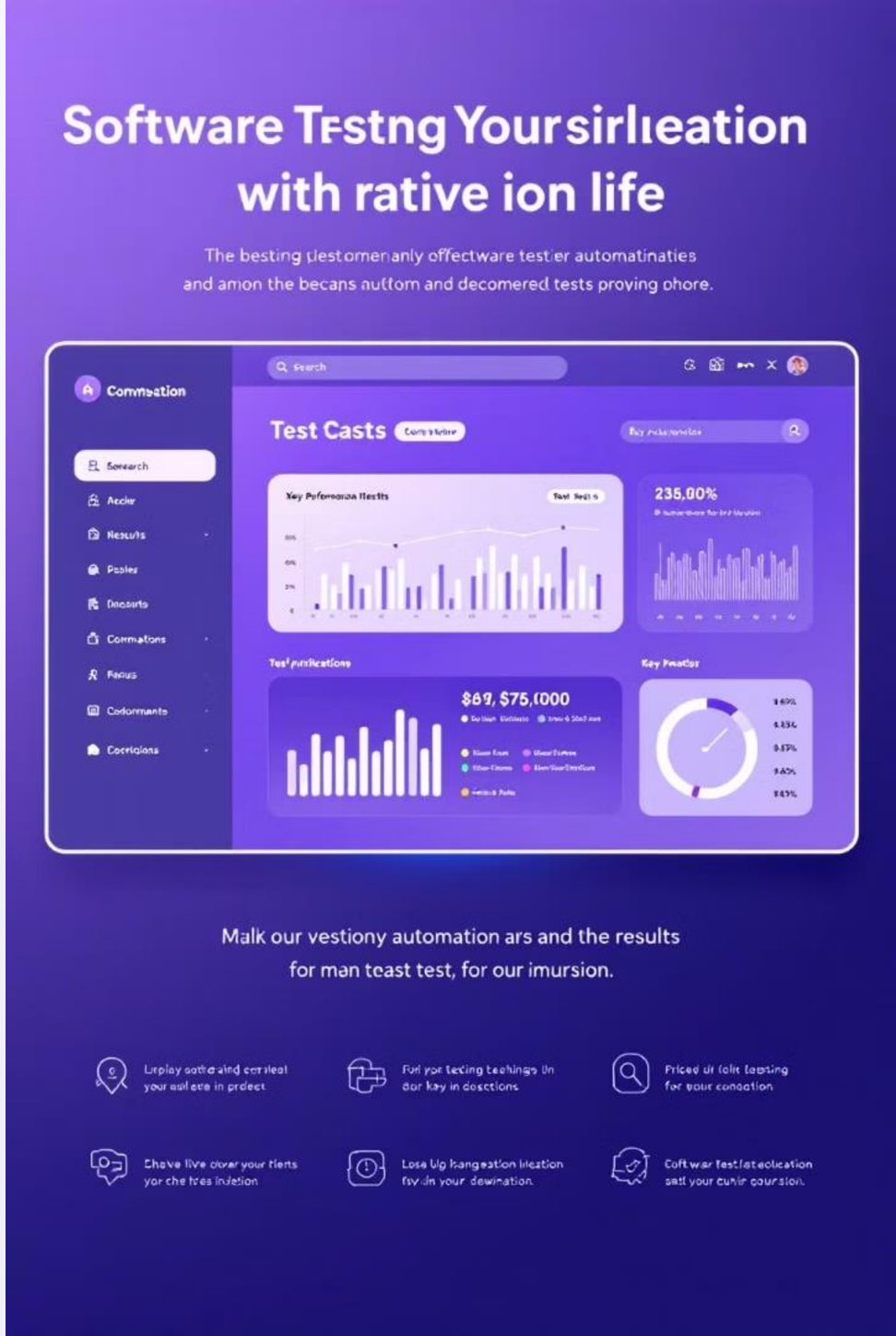
Runs thousands of tests in minutes with detailed error reporting.

Seamless Integration

Works with your existing DevOps pipeline and development tools.

Comprehensive Analytics

Identifies testing gaps and prioritizes critical issues automatically.



Versa Test Automator

Fine-Grained Testing

VersaTest allows testing down to individual message level. Pinpoint exactly where failures occur.

Automated Validation

Automatically generates test messages and validates responses. No manual intervention needed.

Comprehensive Reporting

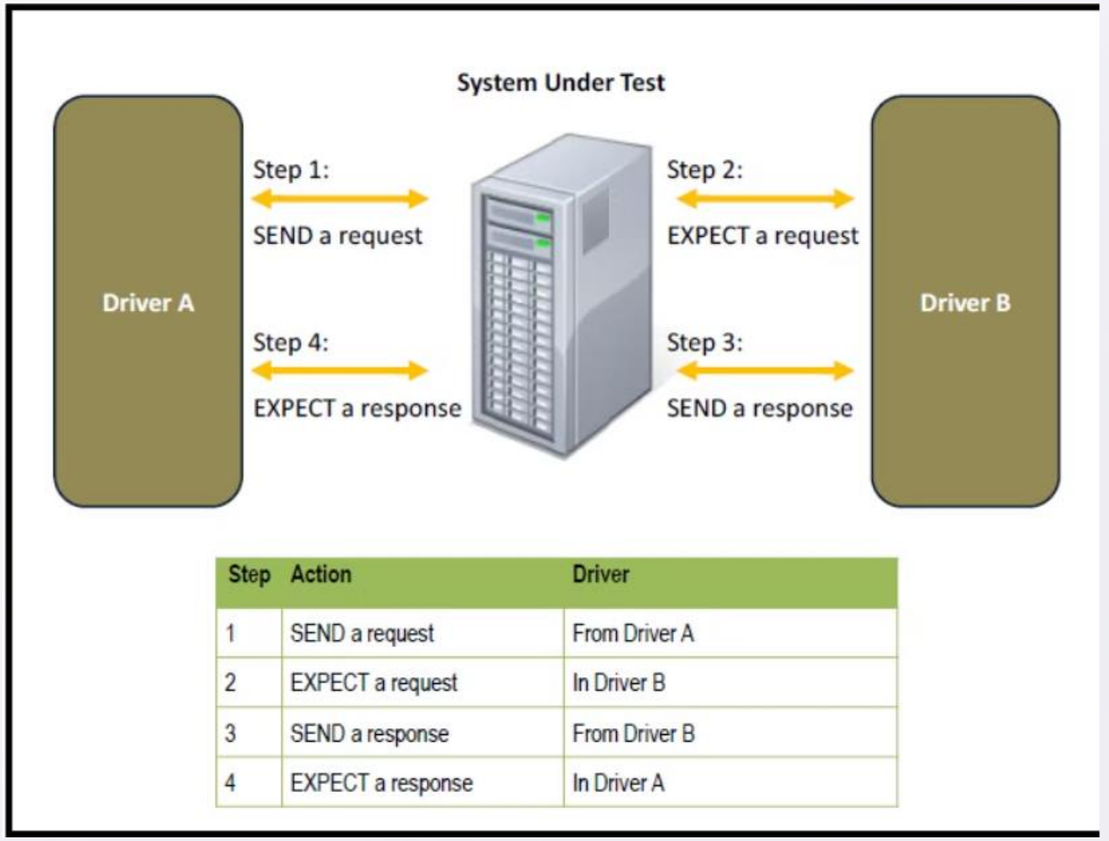
Test logs and results capture automatically. Everything stored as attachments to test runs.

Pass/Fail Clarity

Simple rules language determines outcomes. Failed tests link directly to specific test cases.

VersaTest/AS Test Steps

- A Test Case consists of one or many Test Steps
- Each Test Step consist of a single Test Action



Step	Action	Driver
1	SEND a request	From Driver A
2	EXPECT a request	In Driver B
3	SEND a response	From Driver B
4	EXPECT a response	In Driver A

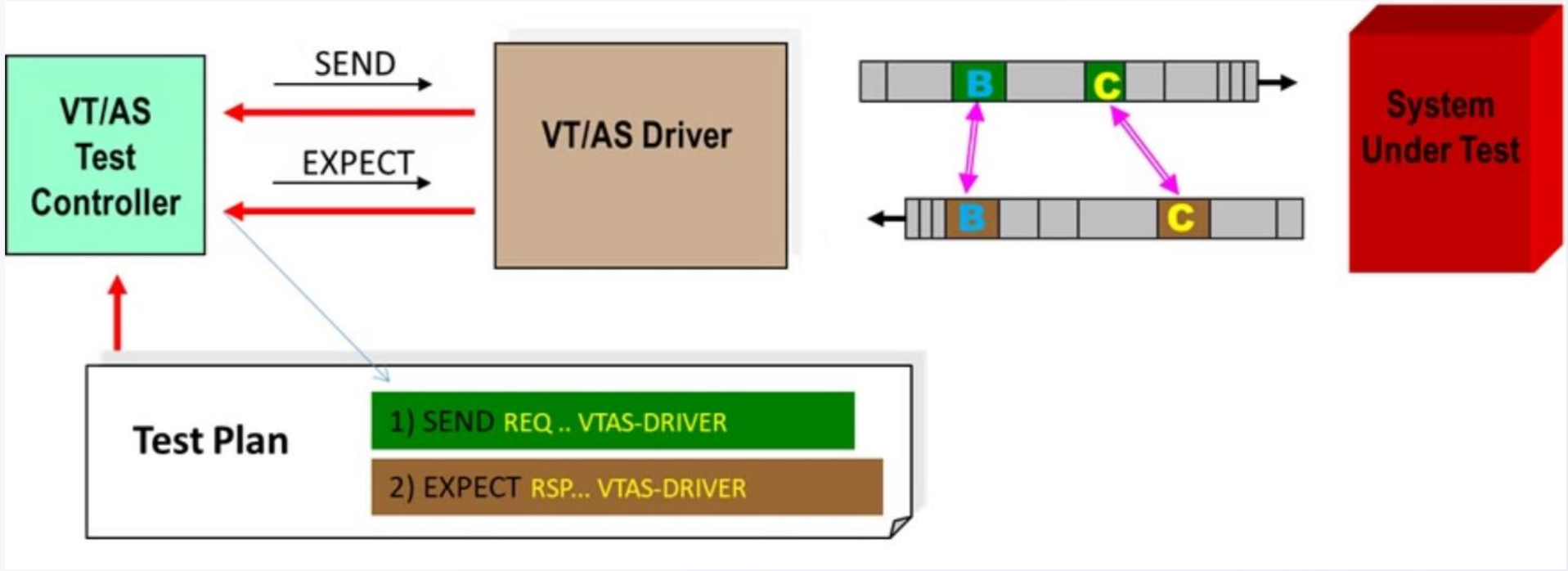
Transaction Generator Mode

VT/AS Test Controller

1. Reads the test cases
2. Sends execution instructions to defined driver

VT/AS Driver

1. Builds and sends the defined message
2. Receives and validates the defined message



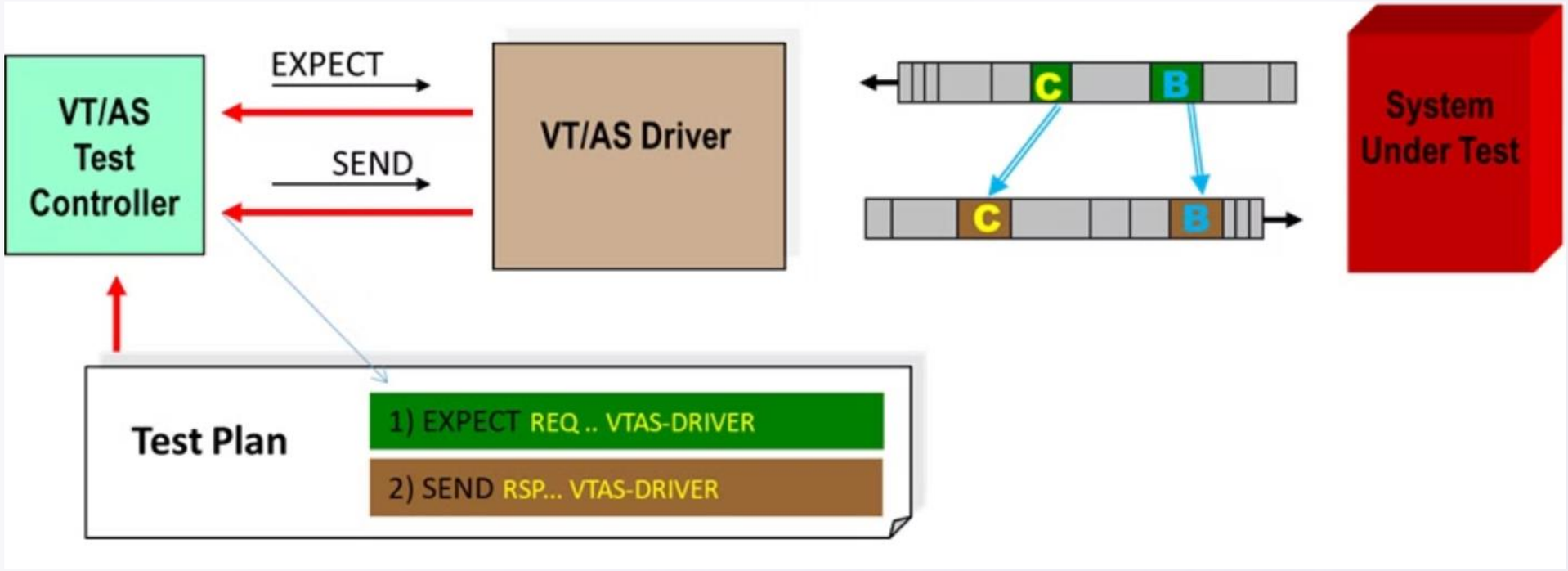
Transaction Responder Mode

VT/AS Test Controller

1. Reads the test cases
2. Sends execution instructions to defined driver

VT/AS Driver

1. Receives and validates the defined message
2. Builds and sends the defined response



Summary of Benefits

Quality & Consistency

Eliminates human error with consistent testing protocols. Identifies defective code quickly and accurately.

Time Efficiency

Automated testing runs up to 50% faster than manual processes. Auto start/stop features save valuable resources.

Cost Reduction

Significant personnel cost savings. Prevents expensive failures by catching issues early.

DevOps Integration

Compatible with Jenkins, Bamboo, and Azure. Simple PowerShell programming connects to CI systems.

For More Information:

www.ascert.com

Rich Greene

RichG@Ascert.com

530-651-4301