

The Necessity for Mainframe Unit Testing

Sam Knutson
Compuware

November 2018
Session BJ

I ♥ MAINFRAME 

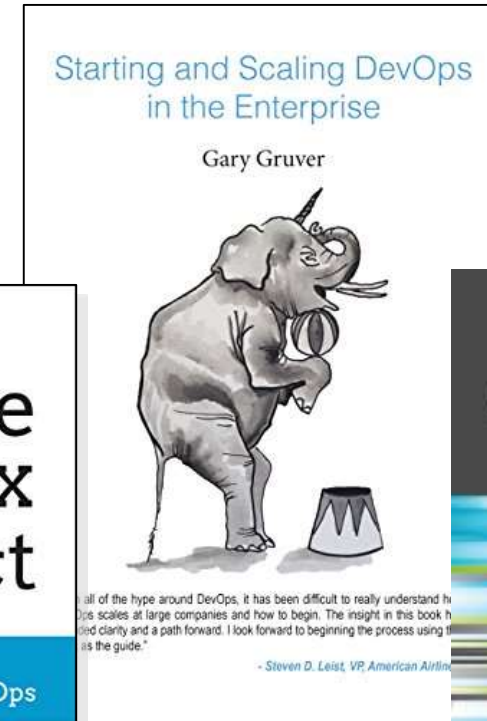
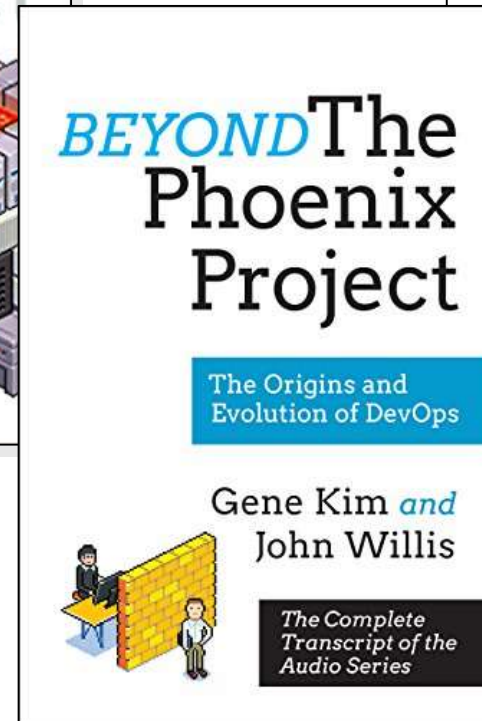
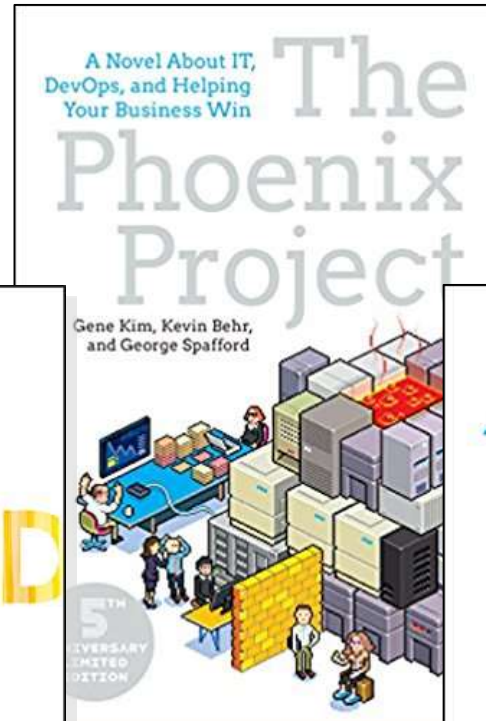
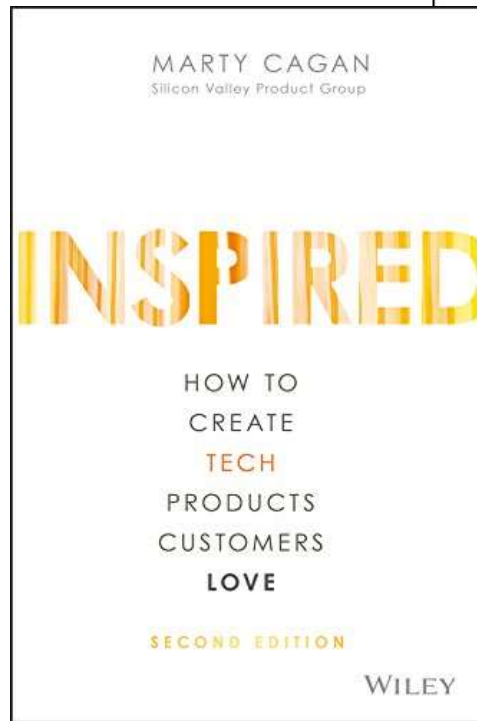


Abstract

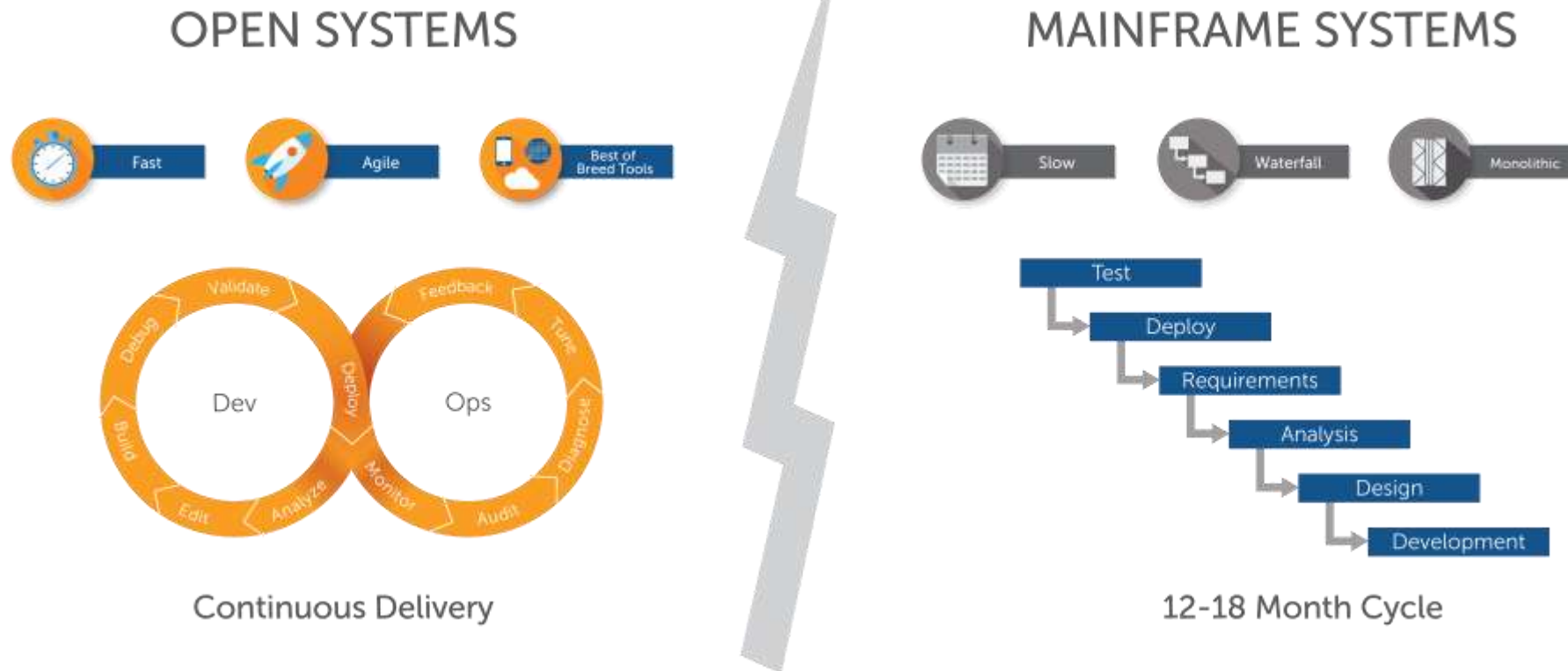
- Automated unit testing is a critical element to mainframe DevOps success, speeding development cycles by eliminating bottlenecks and providing assurance that code changes don't negatively impact another part of a program. Automated Unit Testing provides value including:
 - Increase velocity to production
 - Eliminate dependency on specialized mainframe knowledge
 - Empower novice developers to validate mainframe code changes with the same speed and confidence as other code
 - Support Continuous Integration and Continuous Delivery
 - Reduce time spent manually writing tests, collecting test data or manually creating data
 - Increase confidence to make large mainframe code change
 - Facilitate regression testing
- Find out how unit testing can be incorporated into your mainframe development process. Every methodology for development can benefit from unit testing. Learn the different techniques for getting developers on board with unit testing and how it fits in your methodology. As Agile and DevOps become more common, it is essential that unit testing is implemented and used to maintain code quality. You'll learn about unit testing methodologies, strategies and benefits.



DevOps Product Management Bookshelf



How do you work with your open systems?



Application developers are the craftspeople of the digital business era...entrenched contributors operating in functional role silos must evolve their skills to build great software.

- The Renaissance Developer Gartner August 2014

Compuware's Mission

Next Generation Tools for the Next Generation of Development

- Mainstreaming the Mainframe:
Blended ecosystem that enables
 - Distributed and mainframe teams to share one culture, one process with leading tools of choice
 - Dev and ops teams to interact seamlessly
 - Elimination of mainframe's esoteric nature, making mainframe development just about syntax
- Commitment to elegant simplicity in design and usability
- Leverage partnerships to make widely-used tools applicable to mainframe



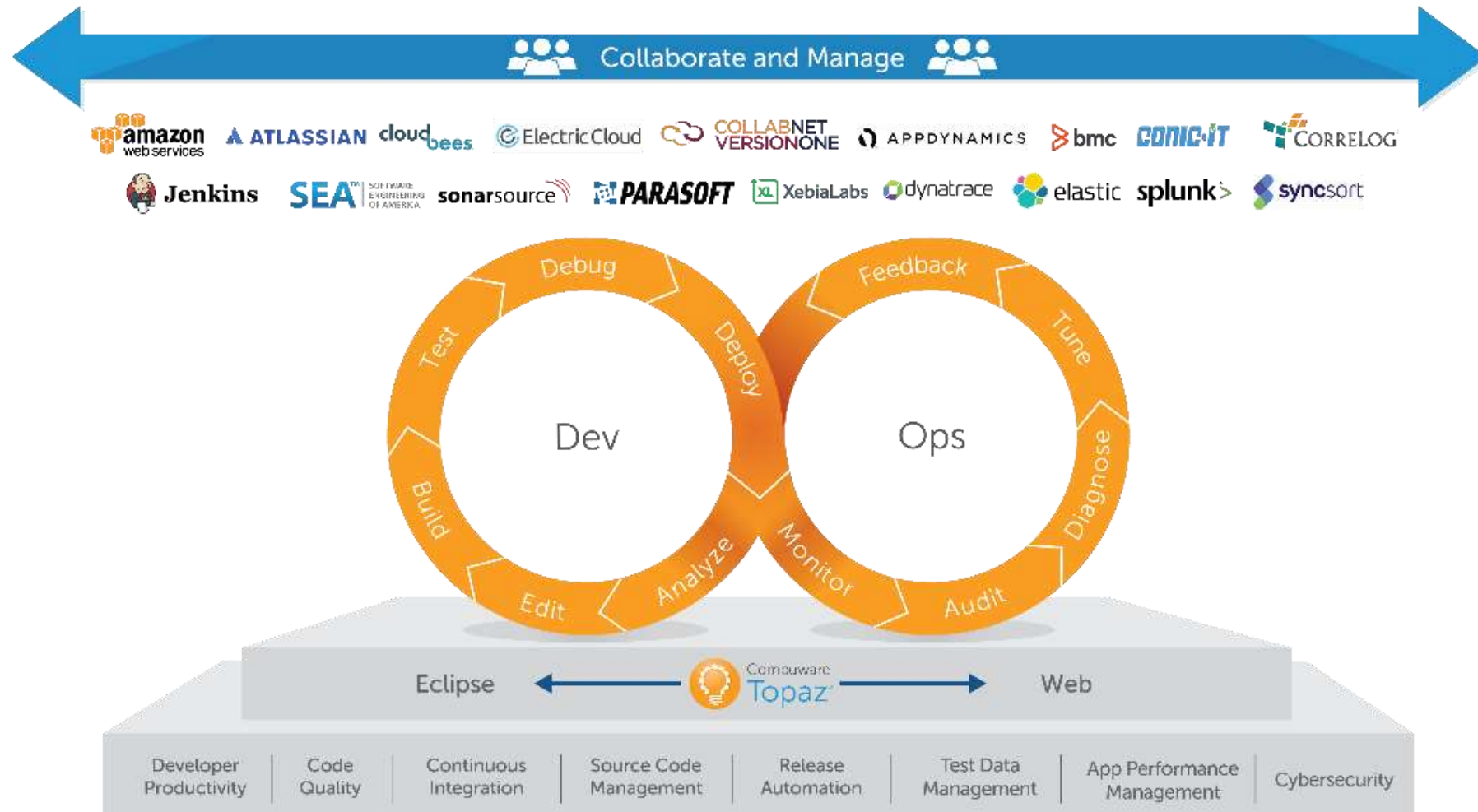
Mainstreaming the Mainframe

Why Enterprise DevOps Should
Take Control of Legacy Applications—Now!



Mainstream the Mainframe

Enabling DevOps Across the Enterprise



Is DevOps and Unit Testing going to come naturally to mainframer's?




 YouTube

Search



Working Code is Gold

- COBOL is Queen!
- Green field vs Brown field
- More than 200B lines of COBOL code
- Maintenance/KTLO vs New Applications



Working Code is **Gold**
and So are the Systems
That Support It

Our simple, elegant solutions bring
Agile DevOps to the mainframe so you can

Keep It That Way



Developers

Need to:

- Understand programs so I know where to put changes
- Debug programs quickly and easily
- Create unit tests and identify testing gaps using code coverage techniques
- Store test results in Sonar where I have a common view of code quality
- Automatically get tests executed when I check in a program with changes





Testing Concepts

Unit Testing

- Act of testing "unit" in your application
- "Unit" is often function or method of class instance
- Unit also referred to as "unit under test"

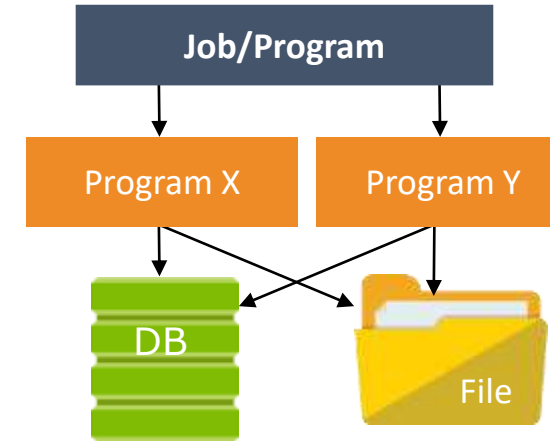
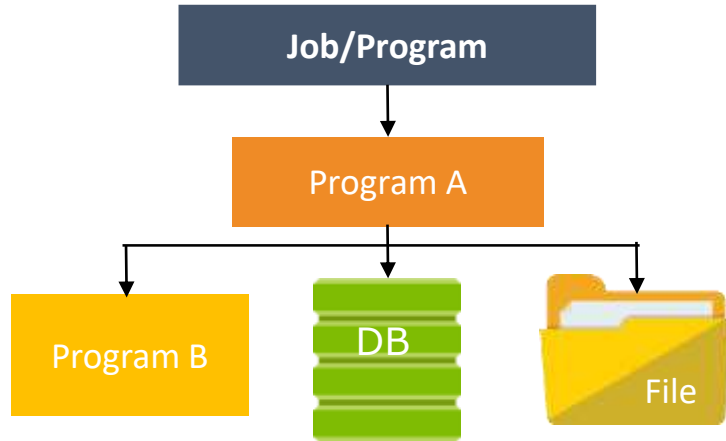
Integration Testing

- Explicitly testing interaction between 2+ "units"
- Verifies application components work together
- Might ensure email was actually sent in integration test

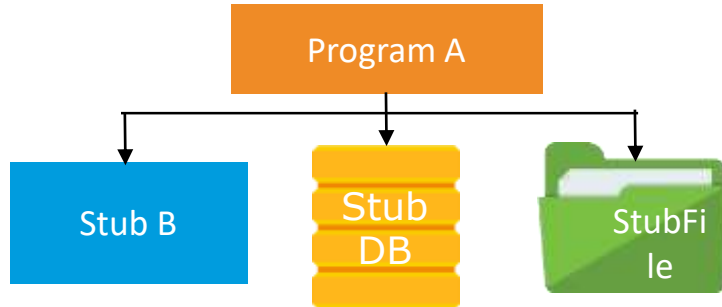
Functional Testing

- Form of integration testing "literally" running application
- Ensure email was actually sent in functional test, because testing code end to end

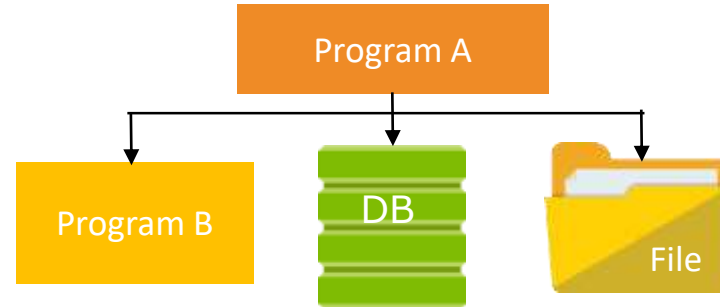
Testing Concepts



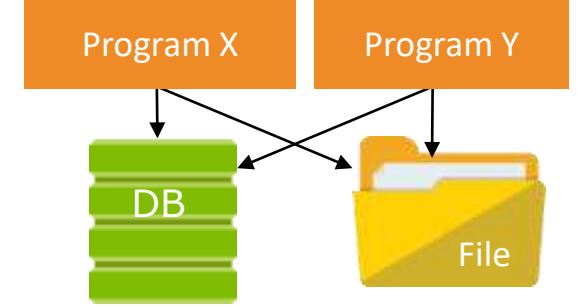
Unit Test



Functional Test



Integration Test



Testing Java

- Compiler, JVM, test framework (JUnit) on local machine
- Direct support in IDE
- CI engines with direct support for Git/JUnit
- Mocking frameworks to stub out dependencies
- Much code work required in creating test case stubs

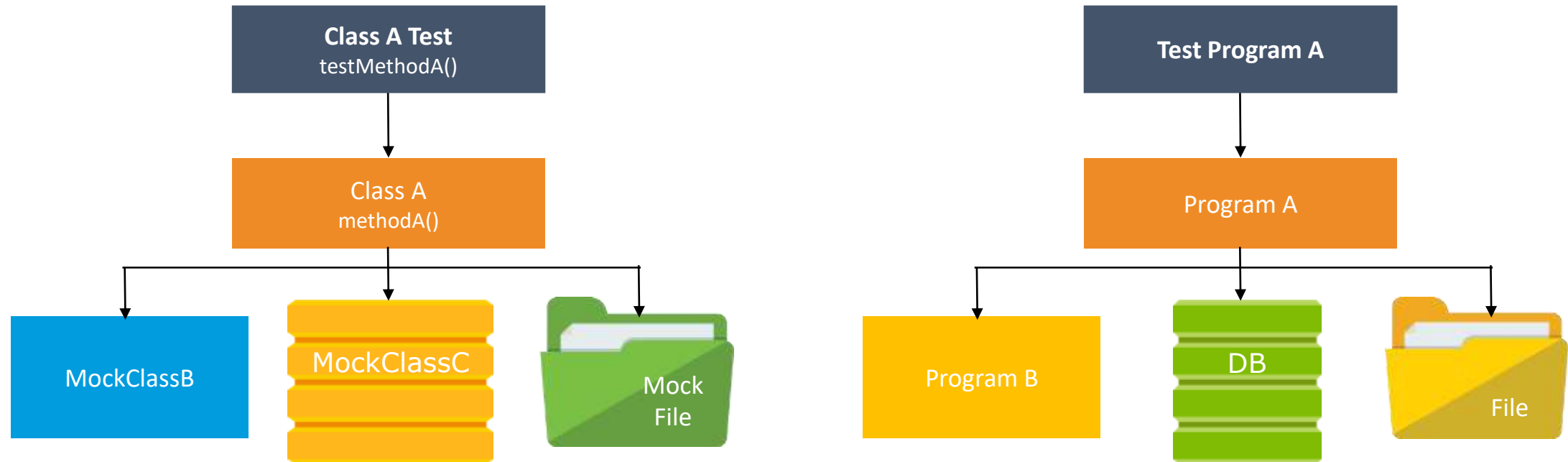


Testing COBOL

- Lacks culture of Continuous Integration (CI) or test
- Must compile and execute on mainframe
- IDE is UI on top of remote connection to mainframe
- Shortage of support from CI engines/orchestration tools
- Non-existent mock libraries/stubbing capabilities
- Like giant Java method; with COBOL:
 - Limited structure
 - Variables are all GLOBAL
 - Limited runtime visibility
- Write, compile and execute COBOL program to test COBOL program—requires program to test program
- Testing a program on the mainframe by providing input and expected output. All sub programs and data access will be live

Java vs. COBOL Testing

- Java and Junit: "Easily" create mock objects and isolate test
- COBOL: Not possible without writing own test framework



Working Effectively with Legacy Code



Legacy code is simply code without tests

Code without tests is bad code. It doesn't matter how well written it is; it doesn't matter how pretty or object-oriented or well-encapsulated it is.

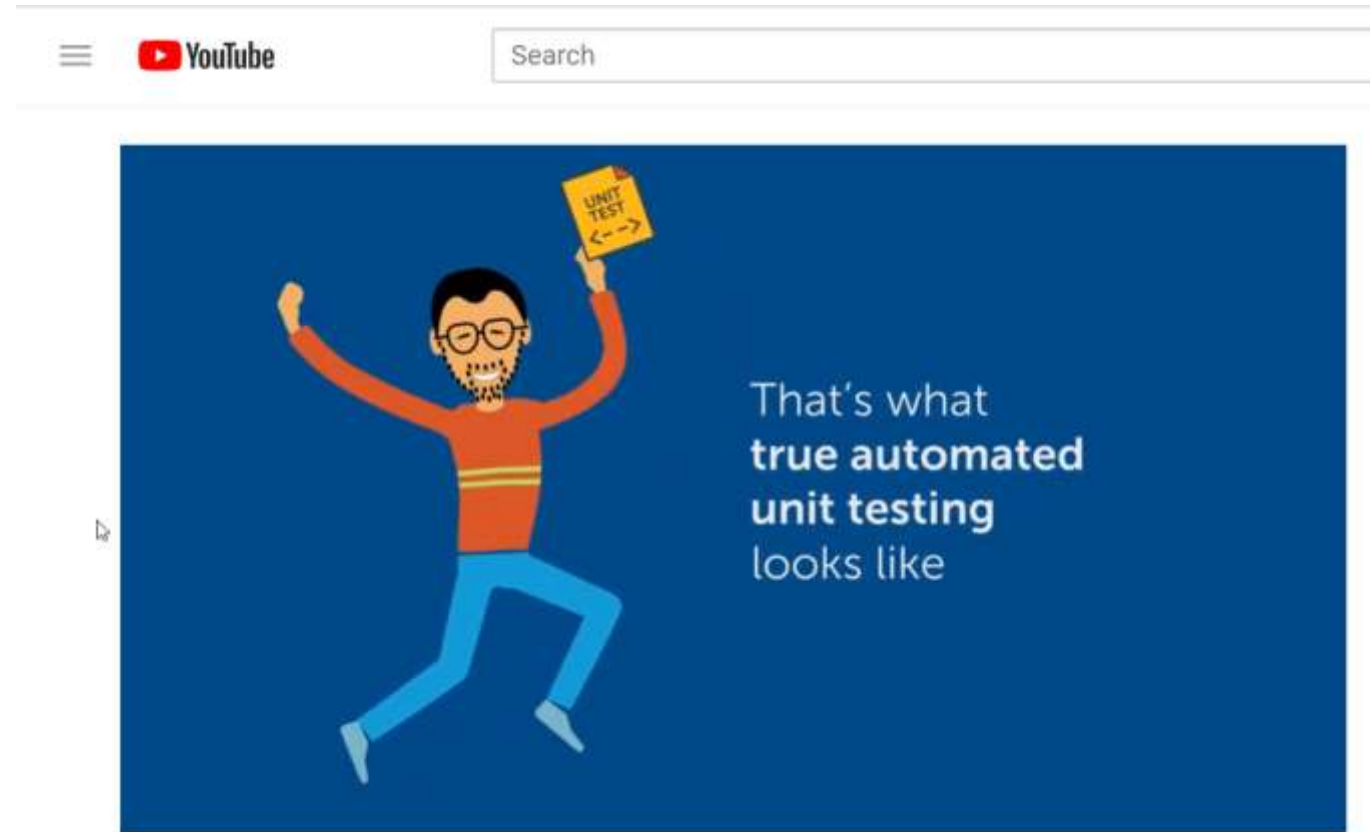
With tests, we can change the behavior of our code quickly and verifiably. Without them, we really don't know if our code is getting better or worse.

- Michael Feathers

“Un-legacy” Your Code

Provide Automated Tests

- Quickly create automated unit tests
- Remove fear of changing code
- Eliminate hassle of moving around test data



How to Unit-test Your Way to True DevOps on the Mainframe

Need Mainframe Program Testing Innovation

- High financial risks of implementing incorrectly
- Speed of mainframe application development can't keep pace with systems of engagement (web, mobile, thick client)
- New workforce—must build expertise
- Suddenly need to work with distributed DevOps teams and leverage APIs



Types of Test Automation

Automatic Test ...

- Creation
 - Data capture
 - Isolation from subsystems
 - Execution
 - Integration with other tooling (Jenkins, SonarQube)
- Case maintenance
 - Update updated when program calculations change
 - Make structure updates for size and type
 - Share definitions and objects for reduced maintenance



High Level of Test Automation Is Required

- **2 billion** lines of new COBOL mainframe code every year
- Programs often grow **very large** (largest so far 600,000 lines)
- Interrelated data = **test data difficulty** (occurs depending on)
- Many different **data formats**
- Many different **subsystems** (Db2, IMS, CICS)
- **Poor** documentation
- **Few** existing **automated** tests
- **30+ years** of accumulated poor coding practices

New Automated Unit Testing Approach

- Rapid COBOL unit test creation
- Interactive test data collection in program context with debugger
- Mock objects/stubs
 - Db2, VSAM, QSAM, CICS, IMS
 - Subprograms stubs
- Easy toolchain integration with Jenkins and Topaz for Total Test plugin
- “Secret Sauce”
 - Completely executable with data, test assertions
 - Easy-to-execute across different test systems



Automated Testing: Distributed vs. Mainframe

JUnit

- Use IDE to generate unit test stub
- Write code into test stub
- Define data for test
- Identify any mock objects needed to compile code and return appropriate response
- Code any test prep into test pre-condition
- Codes any test tear down in test post-condition
- Adds unit test into test suite
- Continuous build process identifies test suite
- Test Runner executes test suite

Topaz for Total Test (TTT)

- Use Xpediter to gather test data call parameters, program results
- **TTT creates complete test case***
- **Unit test uses data stub created by TTT***
- **TTT generates data and program stubs***
- **TTT Runner allows easy on/off of stubs***
- **TTT Runner cleans up after test***
- **TTT adds unit test to test suite***
- Continuous build process runs test suite via CLI
- Test Runner executes test suite

*Automated by Topaz for Total Test

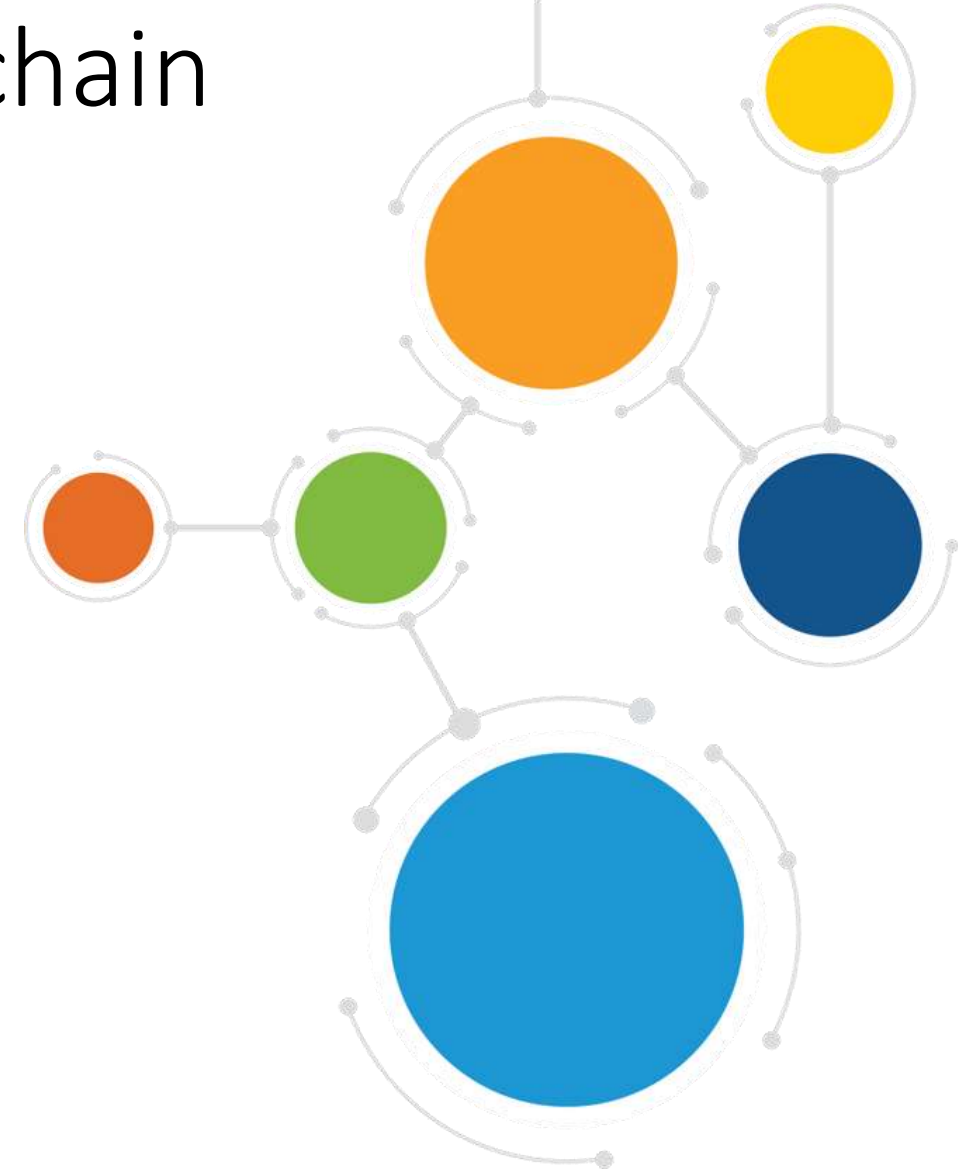
- Unit test
- Automatically create test suites, scenarios, cases
- Central control of test execution
- No invasive code changes to COBOL programs
- Automatically create test assertions:
 - Program input/output parameters
 - Db2 Insert/Update/Delete
 - IMS ISRT/REPL/DLET
 - VSAM/QSAM Write
- Tests can move across system since data stubs move with test
 - Stub data
 - Stub subprograms
 - Stub stored procedures



- Functional test (load modules)
- Create test case for program (Enter input and expected output)
- UI to view and modify data
- Advanced scripting capabilities
 - Automate test setup, teardown and validation
 - Domain-specific language to execute "things" on mainframe
 - Combine "things" in sequence, e.g.:
 - Insert/select data from database, send/receive MQ messages, submit job
 - Iterate over each row
 - Verify output against expected values

Include Topaz for Total Test and XaTester in DevOps Toolchain

- Jenkins and command line interface integrate tests with CI/CD
- APIs execute test suite and cases
- Xpediter Code Coverage integration
- SonarQube dashboard integration



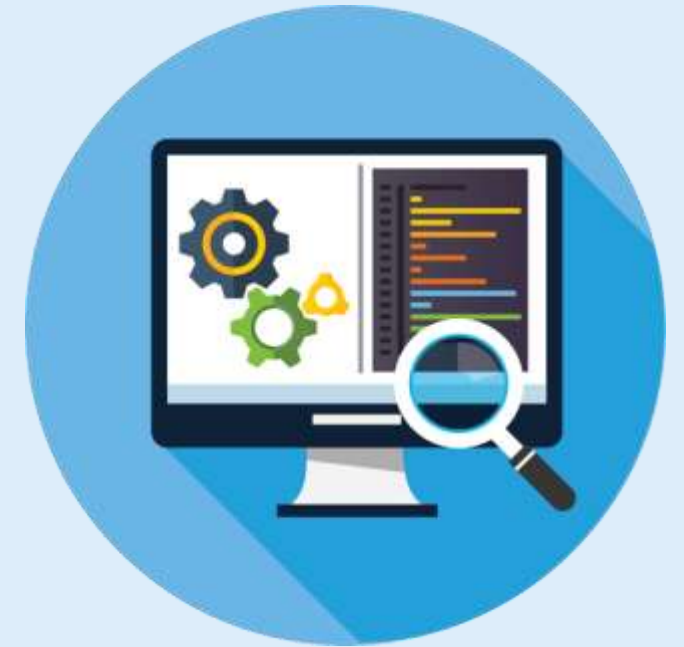
Support for Different Testing Roles

- Agile testing versus waterfall testing
 - QA embedded in Agile team
 - Test group in own silo
- Agile developer writing unit test
 - Rich client for complex tests
- Business analyst doing requirements validation
 - Web interface for simple tests

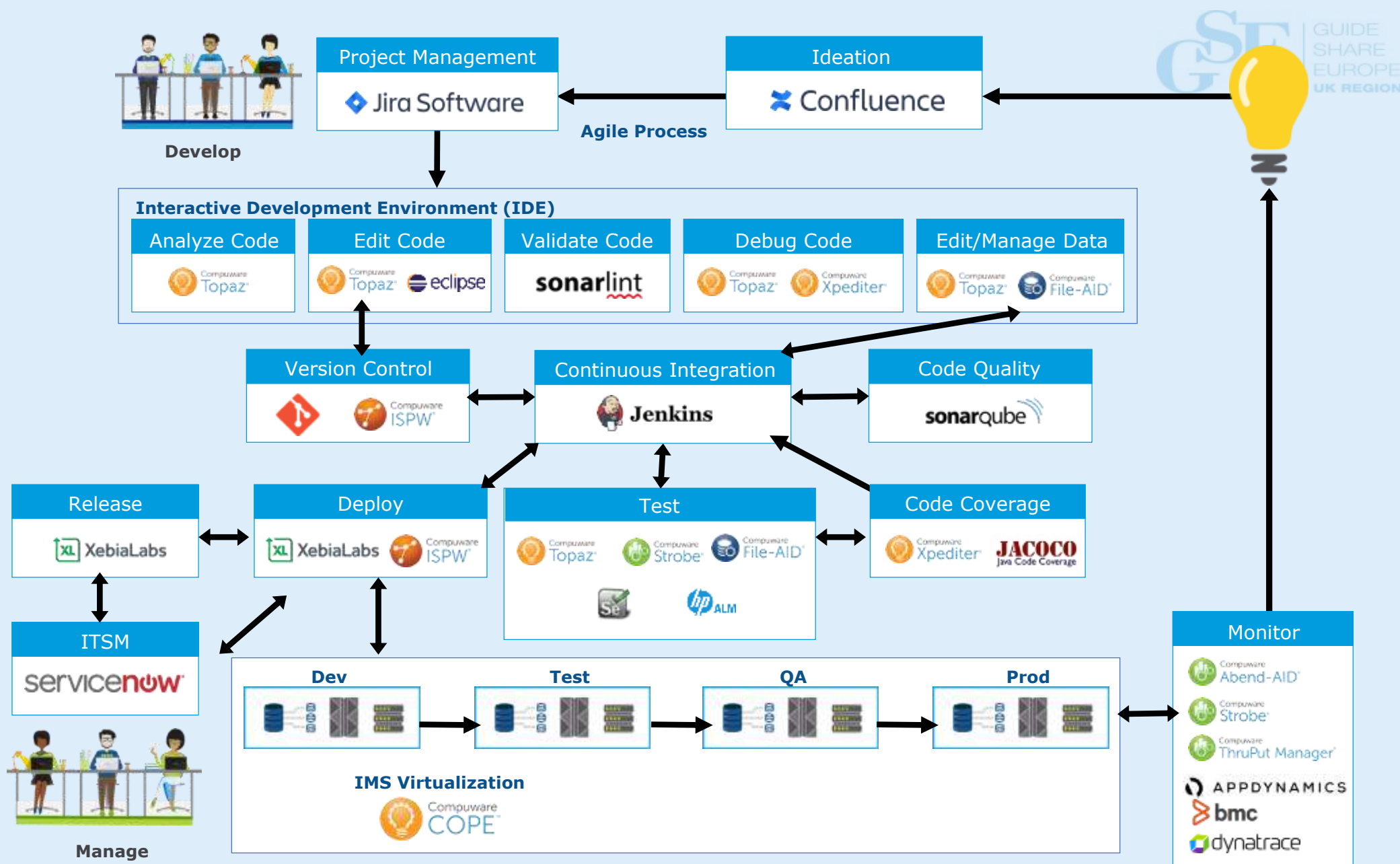


Extending Unit Tests to Functional Tests

- Unit tests isolate calls to subsystems like databases and transaction servers
- Unit test isolate test to single COBOL program and stub out subprogram(s)
- Change unit test to functional test:
 - Remove all program and subsystem stubs
 - Run main program with all required subprograms in execution path



DevOps Toolchain



We want your feedback!

- Please submit your feedback online at
 - <http://conferences.gse.org.uk/2018/feedback/bj>
- Paper feedback forms are also available from the Chair person
- This session is BJ



THANK YOU!

Session BJ



Sam Knutson
Vice President, Product Management

+1 313-227-7604
samuel.knutson@compuware.com
linkedin.com/in/samknutson
@samknutson



One Campus Martius, Detroit, MI 48226, USA www.compuware.com

➤ <http://conferences.gse.org.uk/2018/feedback/fc>