



*Scientia Prudentia et Valor*

# Scientific Test and Analysis Techniques (STAT) for Automated Software Test (AST)

Jim Simpson, Jim Wisnowski, Tom Pestak, Darryl Ahner

ITEA Annual Conference

4 Oct 2017

# What is Automated Software Test (AST)?

*Scientia Prudentia et Valor*

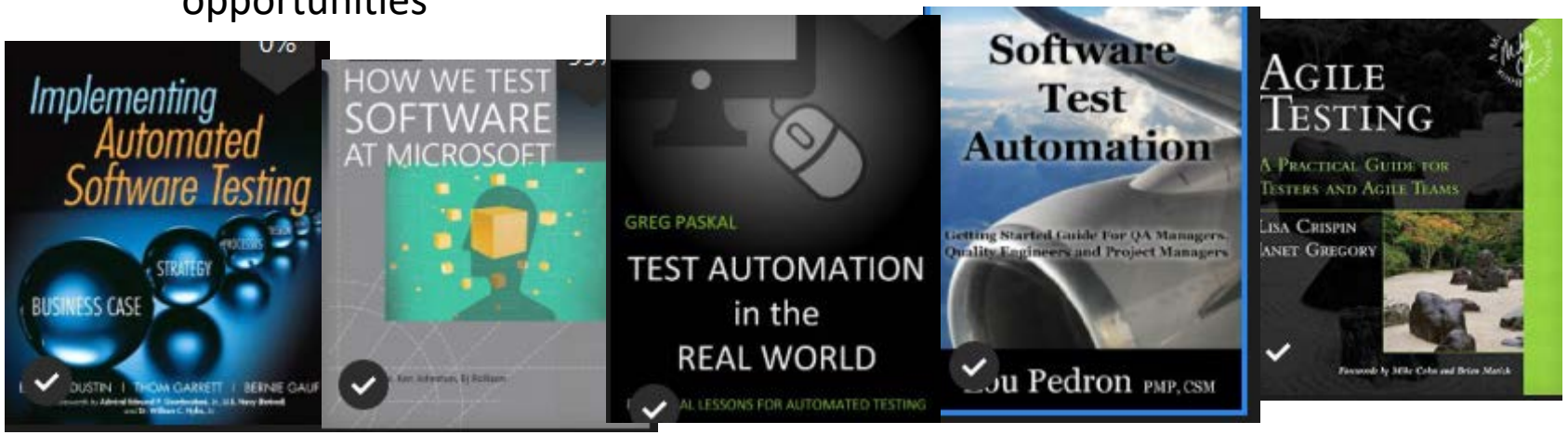
- "The application of automation technology throughout the software testing lifecycle wit the goal of delivering capability faster and more affordably." *Innovative Defense Technologies*
- Important aspects of AST
  - Applies to software-only or software-intensive systems
  - An alternative to manual testing
  - Requires specialized software to control the input feeds, execution, and data collection of the test
  - There are various levels of automation complexity
  - Automation can be applied in all phases of software development and testing. Types include unit, software integration, functional and performance tests
  - The software tools consist of open source and commercial products, that work can operate on the front-end (GUI) and the back-end (objects) of the system
  - Automation capability has both costs (licensing, expertise, automation scripts) and benefits (faster execution, more testing)
- Without automation, testing is limited to what we can do with scarce manual testers

"automaton"

# Project Background

*Scientia Prudentia et Valor*

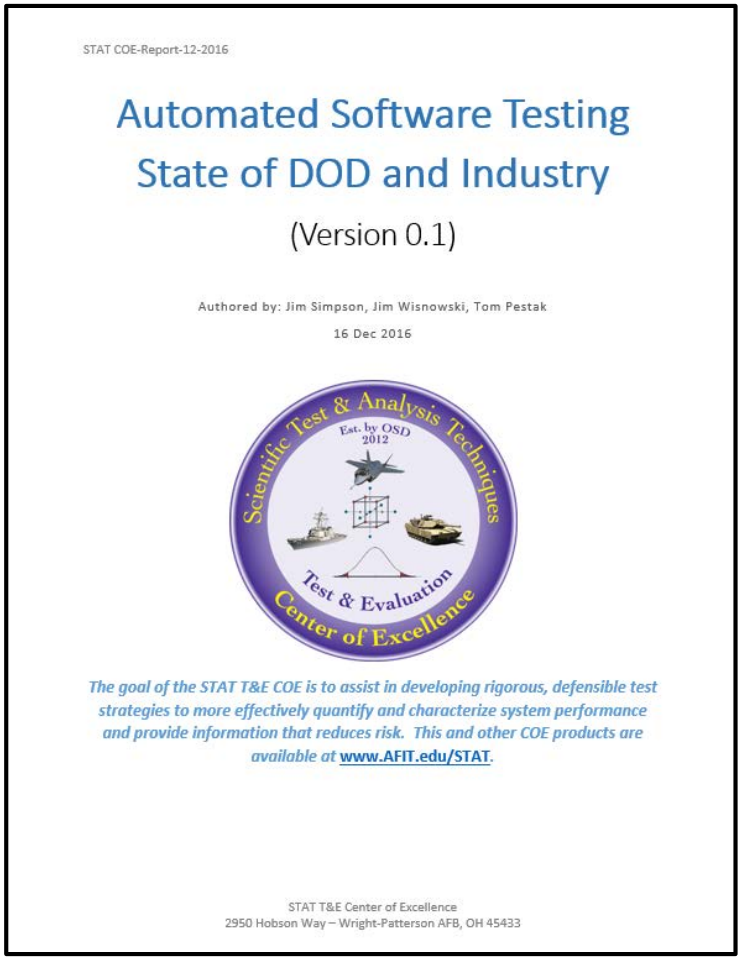
- OPNAV N94 sponsored effort to evaluate software test and automated software test methods
- Objectives
  - Propose general framework for automated testing with quantifiable metrics
  - Identify software test best practices across DOD and industry
    - Focus on C4ISR, system of systems, and autonomous systems
  - Further advance STAT methods such as combinatorial testing
  - Collaborate with DCGS-N and MQ-25 to identify software test enhancement opportunities



# State of DOD and Industry Report

*Scientia Prudentia et Valor*

- Objective: document the current activities of organizations across DoD and industry
- Extensive interviews across all services
  - Primarily face-to-face with demonstrations
  - Identified several outstanding units
  - Most would benefit from an enhanced AST program
- Literature review, interviews, and conferences for state of industry



STAT COE-Report-12-2016

Automated Software Testing
State of DOD and Industry

(Version 0.1)

Authored by: Jim Simpson, Jim Wisnowski, Tom Pestak

16 Dec 2016

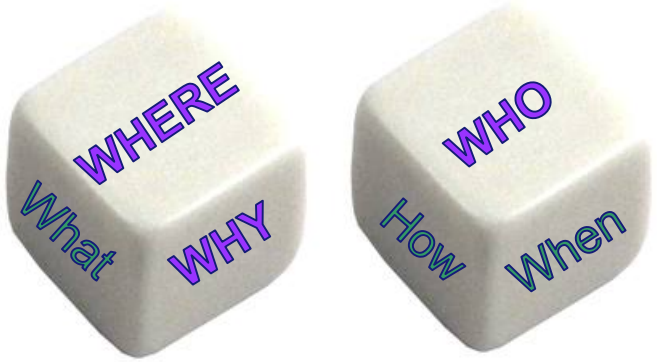
*The goal of the STAT T&E COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at www.AFIT.edu/STAT.*

STAT T&E Center of Excellence
2950 Hobson Way – Wright-Patterson AFB, OH 45433

# Current State of AST in DoD and Industry
## A Perspective

*Scientia Prudentia et Valor*

- **WHERE** – not physically but in relative maturity, how advanced is the DoD? Primary report emphasis
- **WHO** – explains the organizations automating, major contributors and industries
- **WHY** – why is the current state where it is? Provides the insights into barriers to progress and the enablers that have AST where it is now




- What – given in textbooks, best practices and here only to articulate the state
- When – now
- How – not the purpose of this report, but can be important to clarify a current state point

# Automation Hurdles

*Scientia Prudentia et Valor*

- Tools?
- People to Automate
- Scripts
- Script Reuse
- Where to apply it
- Time to get ready
- Return on Investment (ROI) study
- Leadership support
- Training
- Changing tools
- Staffing
- System under test (SUT), OS, hardware changes

# Organizations Visited



- Navy: SPAWAR Pacific and Atlantic: HQ 5.9, DCGS-N, NITES-Next, MTC2, SSC-Atlantic DCGS-MC and 5.9, MQ-25, Common Control Station, JMPS, F-35, MQ-25, Corona Performance Assessment Division, NSWC Dahlgren
- Air Force: DCGS-AF, AFLCMC, PEO C3I, 46 Test Sqdn, AFRL, AFOTEC, SMC, PEO BES
- ARMY: DCGS-A and ATEC
- Marine Corps: MCOTEA, MCTSSA, and DCGS-MC, GCSS
- OSD and Government: IDA, NIST
- Academia: Arizona State University, UMass Dartmouth, George Mason University
- Industry: Silverthread, PCI, Jackpipe, Innovative Defense Technologies

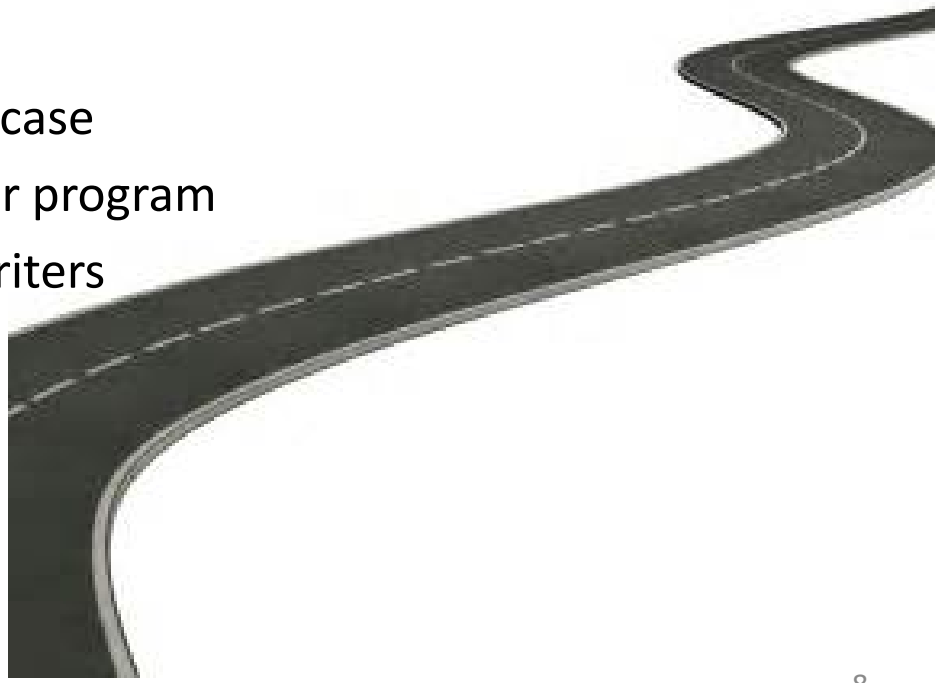
- For each Service, what is happening in terms of:

| | |
|---|---|
| Policy and Guidance | AST Initiatives |
| Culture for AST | Notable Programs using AST |
| Centers of Excellence | AST Tools Experience |
| AST Metrics | Test Space Coverage |

# Success Pioneers

*Scientia Prudentia et Valor*

- Some organizations and programs have made great strides
- 46TS, DCGS-N, DCGS-A, GCSS-MC, MTC2, MCTSSA and Link-16, JMPS-AF, JMPS-N
- Common Denominators
  - Set the goal and stick to it
  - Build and market the business case
  - Choose tools that work for your program
  - Seek and find or mold script writers
  - Start small and grow capability

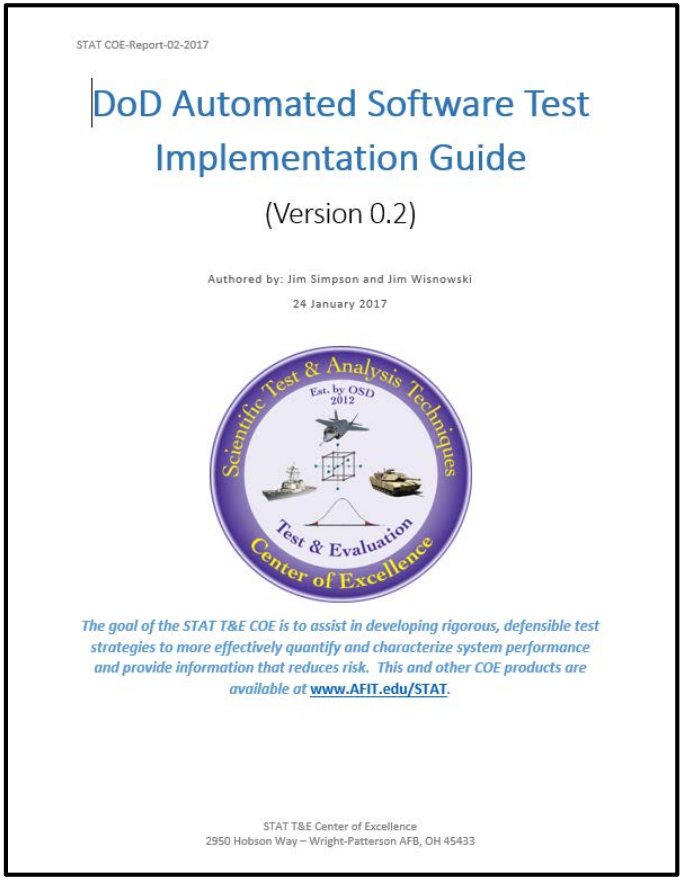
# Valuable Lessons Learned

*Scientia Prudentia et Valor*


1. Important to **not rely on record/playback tools** - only to start basic functionality and proof of concepts.
2. Automation process needs to be **treated as a software development effort** and tests must be *designed* from that perspective.
3. **Staff** the project **with software developers** familiar with techniques Java, JS, HTML, xml.
4. AST project, **test scripts need to be designed**. Automate manual test cases without designing modularity will not work.
5. **Allow for maintenance upfront**. Framework, scripts, and tools all need to be maintained and factored in for significant resources.
6. The **best candidates for automation for ROI are regression tests**, back-end tests

# AST Implementation Guide

*Scientia Prudentia et Valor*



STAT COE-Report-02-2017

DoD Automated Software Test Implementation Guide

(Version 0.2)

Authored by: Jim Simpson and Jim Wisnowski

24 January 2017

*The goal of the STAT T&E COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at www.AFIT.edu/STAT.*

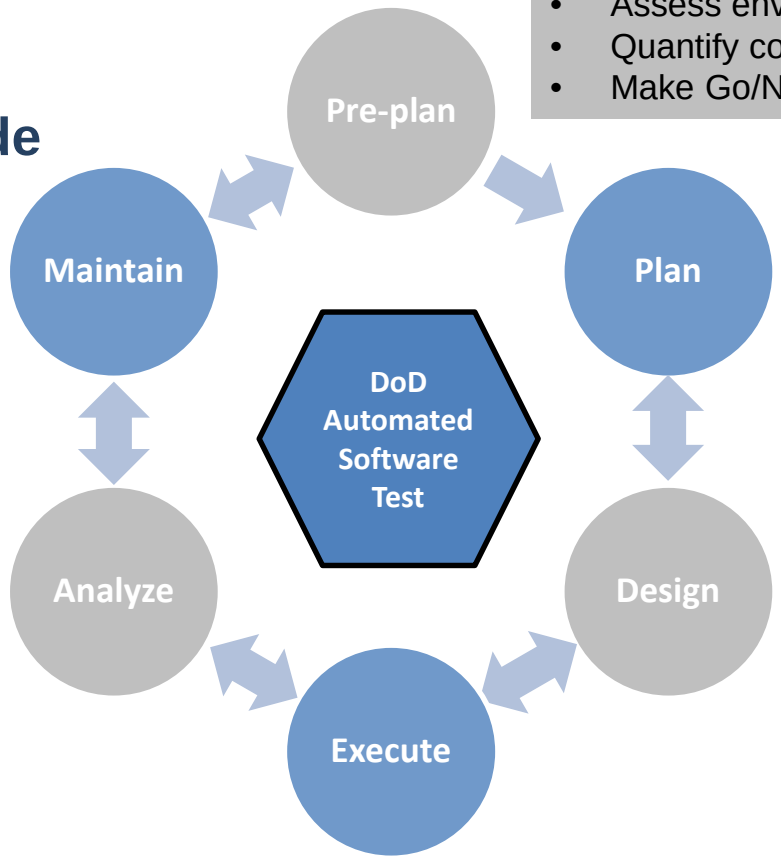
STAT T&E Center of Excellence
2950 Hobson Way – Wright-Patterson AFB, OH 45433

- Guide for managers and practitioners focused on tasks required for a successful AST program
- Overarching theme of Return-on-Investment for automating specific tests
- Step-by-step methodology broadly applicable to DoD testers across the Software Development Life Cycle

# Implementation Guide Overview

*Scientia Prudentia et Valor*

## DOD Automated Software Test Implementation Guide




- Assess environment, system, manpower
- Quantify costs and benefits;
- Make Go/No-Go decision

- Identify automation requirements
- Prioritize requirements
- Identify and compare tools
- Determine automation framework needs
- Outline test scripts
- Create test plan

- Select tools / test platform
- Generate test scenarios
- Determine test cases
- Establish data processing
- Create configuration control
- Conduct design review

- Create manual tests from users' operational profiles
- Develop and refine automation scripts
- Integrate tools within automated test suite
- Verify automation with smaller test cases
- Execute test cases and scenarios and contingencies

- Establish data output
- Analyze anomalies
- Check repeatability
- Update metrics and compute ROIs
- Refine AST solution

- Configuration management
- Update code
- Manage script repository
- Track defects and assess trends

# Pre-Planning Phase



- Key Elements
  - **Research**, invest time and gather information for making an informed decision on automation
  - Perform **cost-benefit analysis** and compute an ROI, and include long-term benefits
  - **Decide** whether or not **to automate** based on research into program test plan, automation capabilities plus opportunities, manpower skillset / resource needs, and quantifying automation costs & benefits
- Deliverables
  - Make the decision: automate or not
  - Quantify gains (time and money) to the program
  - Compute the automation return on investment for the short term (1 year) and longer term (2-4 years if appropriate to the program)
  - Document all analyses

# Planning Phase



- Key Elements
  - Develop an automated software test plan by identifying
    - and **prioritizing test requirements**
    - and **assessing appropriate automation tools** with quantifiable and discernable metrics
    - **barriers to automation implementation**, drafting the test automation **framework**, and outlining the test **script needs**
- Deliverable: Automated Software Test Plan
  - Detailed system description
  - Delineated requirements along with the DEF decomposition matrix down to automatable tasks
  - Prioritization methodology and ranking
  - Planned AST framework
  - Candidate tools along with capability evaluations
  - Responsible POCs for critical tasks
  - Test resources
  - Timelines

# Analysis Phase



- Key Elements
  - Focus on **output of automated test**, typically involving recording, data files or log files
  - Purpose is to **combine, manipulate and analyze the output data to learn** output errors, integration issues and faults of the SUT
  - Individual steps include setting the **data format**, **assessing** the output data, ensuring **anomalies** are real and characterizing them, revising automation metrics and ROI
- Deliverable: Compute ROI and Consider Future Options
  - Go deeper with the current effort
  - Scale up with more user load
  - Automate new requirements
  - Adopt different tools
  - Transition from GUI to code-based API tests
  - Conduct contingency test cases
  - Abandon automation

# Maintenance and Sustainment Phase



- Key Elements
  - Typically the **most time consuming** of automation
  - Once test scripts written, executed and refined for optimal use, **something changes** (e.g. SUT, test environment including monitors or operating system versions and IT patches, automation tool version)
  - **Scripts** will fail to execute properly unless **revised**, which is a primary task in the maintenance phase
- Deliverables
  - A plan for future automation success by knowing the maintenance process and workload
  - Manage the library of functioning scripts – update and revise
  - Consider tool upgrades or alternatives based on goal to improve the maintenance situation

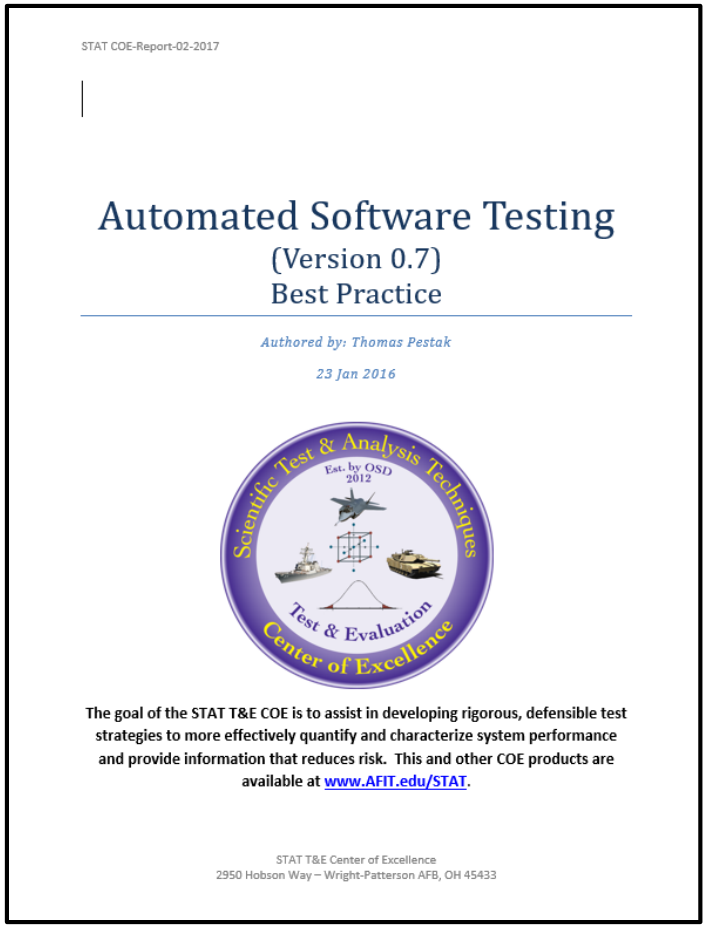
# Automated Software Testing Metrics



- Return on Investment
  - Automation software license cost
  - Automation software training and ease of use
  - Development time
  - Maintenance time
  - Additional hardware/firmware required
  - Risk of failure to effectively automate
  - Reuse of script/procedure potential
  - Improvement in defects (critical, not mission critical) identified and removed
  - Improvement in coverage
  - Shorter software development cycle
- Testing time and frequency
- Analytic burden
- Increased confidence
- Scalability

# AST Best Practice

*Scientia Prudentia et Valor*



STAT COE-Report-02-2017

Automated Software Testing
(Version 0.7)
Best Practice

*Authored by: Thomas Pestak*

*23 Jan 2016*

The goal of the STAT T&E COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at www.AFIT.edu/STAT.

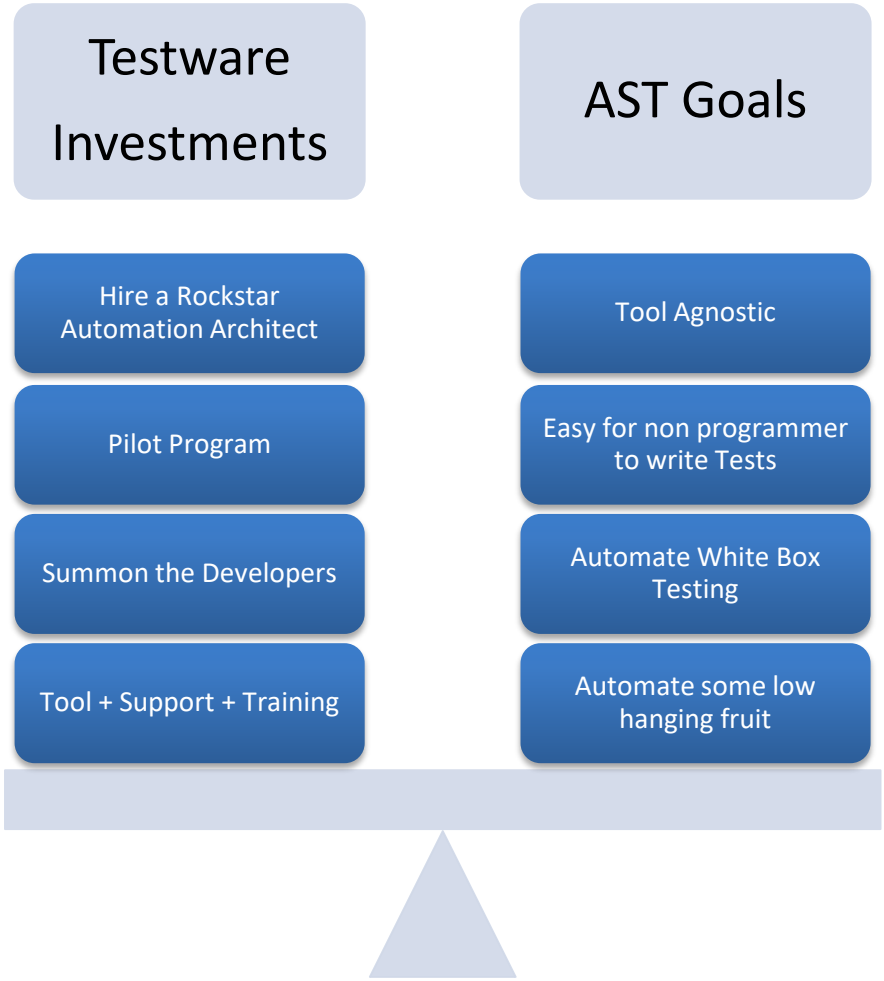
STAT T&E Center of Excellence
2950 Hobson Way – Wright-Patterson AFB, OH 45433

- Identifies numerous best practices adopted from DoD and industry
- Describes non-technical issues that inhibit successful implementation AST
- Articulates the decision making process for strategizing automation based on the test phase
- Defines skill sets required to implement AST
- Highlights best practices for developing a testware architecture
- Discusses major commercial, open source, and DoD (ATRT) tools

# Best Practice

*Scientia Prudentia et Valor*

- Executive Summary
- Background
- AST Goals and Traps
- What Can Be Automated?
- Roles and Skillsets
- Tips for Good Testware
- Tools for AST
- Conclusion
- Further Reading
- References

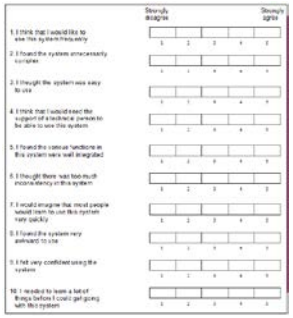


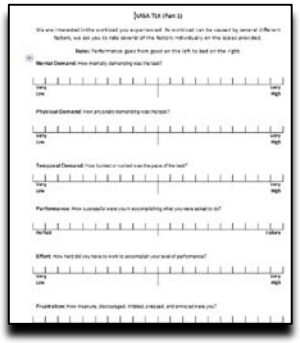
# Program Support

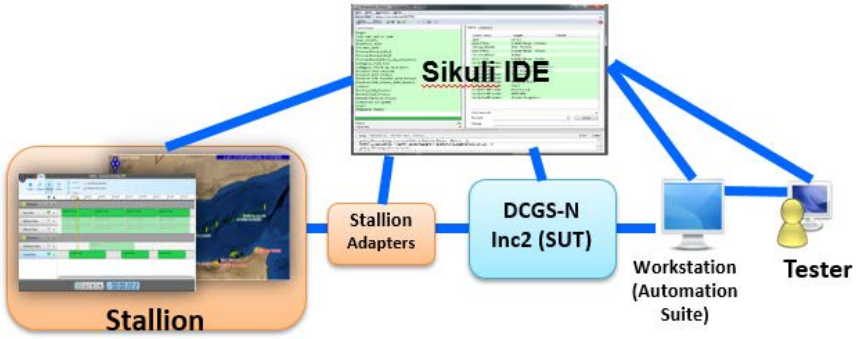
*Scientia Prudentia et Valor*

- PMA 281 Common Control Station and Joint Mission Planning System (JMPS)
  - Provided test designs, integrated training, conducted surveys, and performed comprehensive statistical analyses for an AST tools comparison study for HP UFT, MS TFS, ATRT, and TestComplete

- Distributed Common Ground Station-Navy Increment 2 and System of Systems
  - Participated in agile development program as they implemented an AST proof of concept







# Summary

*Scientia Prudentia et Valor*

- Thanks to OPNAV N94 for funding this outside look at AST with a STAT lens
- Numerous interviews, visits, demonstrations, and literature reviewed captured in State of AST in DOD and Industry report
- AST Implementation Guide v1 describes specific tasks to accomplish for preplanning, planning, design, execution, analysis, and maintenance phases
- Best Practices report provides insight into common pitfalls and offers guidance for success
- Program support provided to DCGS-N and JMPS provided value to organizations and important input to the AST documents
- AST requires a culture change and management commitment to effectively apply these methods as part of routine software test
- Many options for follow-on work—DoD has much work left to do in AST
- Phase II begins next month

# BACK UPS

# Design for Automation Phase

*Scientia Prudentia et Valor*

- Key Elements
  - take the automation plan another level deeper in detail and make decisions for how best to execute automation. In this phase automation tools are selected and made available, test scenarios for automation are generated, test cases are determined, the output analysis strategy is designed, and configuration control is established, all culminating in a design review
- Deliverable: Design Review
  - Demonstrate readiness for automation execution.
  - Include a brief summary of the pre-plan decision process and evidence for the automation decision
  - Plan should be covered in detail and highlight any areas of concern
  - Design covers the initial capability to generate prototype scripts Revisit the ROI before committing the additional resources

# Execution Phase



- Key Elements
  - the activities and decisions that enables a test otherwise to be conducted manually, to be automated. It often starts by interviewing a system operator or capturing the manual tester's steps, then decide the best automated test environment, integrate the tools within the designed test framework, develop and refine the automation scripts, and iteratively test out the execution while refining the process.
- Deliverables: Automation execution and output generated

# Pre-Planning Phase

*Scientia Prudentia et Valor*

- Conduct research on program/system and AST
  - Acquire high level understanding of system and AST from similar programs, vendors, texts; reach out to DOD AST experts
- Understand your environment
  - Much of DOD not in AST mindset; requires strong leadership for cultural change; Agile and DevOps test driven development; be aware of resources required for manpower, software, hardware, hosting, cybersecurity, cloud computing
- Assess manpower and skill requirements
  - Understand differences between testers, automators, developers, system engineers; hire or grow internally; long term versus short term requirements; are right pieces in place for coaching/mentoring?
- Determine potential benefits from automation
  - Software quality improvements, decreased test time, deeper coverage, increased reporting, freeing up critical manpower for higher priority efforts, better scale/load testing
- Quantify costs of automation
  - Short term vs long term investment, fixed vs variable costs, software licenses, staff, tool familiarization time
- Decide based on expected Return on Investment
  - Cost-benefit with sensitivity analyses, account for risk especially if first AST, future AST options

# Planning Phase



- Identify automation requirements
  - Determine functional requirements of system and non-functional (operating system, hardware, IA); find an AST pro to discuss options; think beyond traditional testing
- Develop DEF requirements prioritization
  - Build work breakdown structure table with developmental objectives, detailed requirements, measures, priority, risks
- Identify and compare tools
  - Not a single tool solution: project management, scenario generation tools, test coverage tools, automation tool, programming tools, IA and security tools, output tools; assess for ease of use, scripting language, test types (GUI, API), support, cost, use in DOD; interview users and experts
- Determine automation needs
  - Match to tool capabilities; code based vs GUI tradeoffs; mimic operational performance by hosting tools outside SUT
- Write and execute test (pilot) scripts
  - Query test script repositories first; start simple and build pilot test cases to evaluate automation environment performance
- Publish automation test plan
  - Document overall approach to AST; assign responsibilities and resources; establish timelines

# Design for Automation Phase



- Determine what to automate
  - Consider software development lifecycle and test phase (DT, OT…); conduct risk analysis along with cost; study similar automation efforts
- Select tools
  - Narrow search to promising candidates; research integration issues; practice with tutorials and vendor solutions; purchase licensing
- Acquire automation capability
  - Develop expertise on tools through collaboration and purchases training, recruit automators, testers, and coders
- Determine automated test platform/framework
  - Decide on operating system, virtual machines, cloud, hardware requirements; modeling and simulation environment (e.g. Stallion) for output; simultaneous users
- Generate scenarios and test cases
  - Scenarios built from multiple test cases;  use coverage tools with data-driven testing, determine level of abstraction (live, recorded live, or simulated)

# Design for Automation Phase (continued)



- Assess expected coverage
  - % test cases executed; determine high risk cases; implement statistical and combinatorial methods; determine methods to implement the combinatoric solution; improve
- Decide test timing
  - After agile sprints (monthly), fleet release capability (quarterly), new software capability (regression), continuously run over night
- Establish data acquisition and processing systems
  - Set system up to capture test output data; run prototypes with negative testing to identify areas where greater detail for log files needed; determine frequency of output; focus on ease of identifying defects and ability to conduct analyses
- Create configuration control construct
  - AST is continuous process with both SUT and automation framework changing often; may not be able to keep up with design changes; keep detailed records of updates and identify POCs
- Conduct design review
  - Demonstrate initial capability; execute prototype scripts; declare readiness for execution

# Execution Phase

*Scientia Prudentia et Valor*

- Capture users' operations of system
  - Software engineer working with system SME to find usage profiles; focus on most likely and highest risk paths; create primary process flows and contingency paths; assess automation potential
- Create manual tests
  - Detail step-by-step tasks to test selected cases; record several trials of manual test; evaluate automation potential; use multiple testers if possible for best approach
- Determine automated test environment
  - Evaluate test execution options: SIL, lab, ship or operational environment, cloud, contractor facility; determine level of abstraction to automate from actual system; know hw and sw requirements
- Integrate tools within automated test suite
  - Likely no single tool will do all required automation, engage software development/coders, set up hardware master/slave system; test communication between tools
- Develop and refine automated solution
  - Write and test scripts in highly collaborative environment; iterative solution with continuous SME input, peer review code and output; perform verification and validation of automation scripts
- Verify automation pilot results
  - Start with small test cases and scenarios; use negative testing; ensure correct output is written to right files using proper formatting
- Execute automation
  - Run automated tests; continuously monitor for errors that stop process; execute contingencies

# Analysis Phase



- Establish data output format
  - Create informative log messages; highlight errors in output files; provide screenshots of system status upon error; ensure output possible for further data reduction and analyses
- Analyze anomalies
  - Determine source of discrepancy whether a system software failure or the automation (Gregg v Greg in GUI); deal with false positives; perform root cause analysis, socialize software trouble reports in collaborative environment; use information to inform future testing
- Summarize requirements tested and identify gaps
  - Check log to see what actually was tested, cross check with covering arrays/combinatorics
- Check repeatability and reproducibility
  - Automation may not be stable especially if GUI capture, exact same test may not produce same results with same tool (repeatability) or different tools may produce inconsistent results (reproducibility); standardize test procedures; start planning for maintenance
- Compute and update automation metrics
  - % lines of code covered, % paths covered, time to execute test, manpower, time to develop test cases, defects per KSLOC, software quality improvement, total test time, reduction in dev time
- Compute ROI and identify future possible automation
  - Decide to go deeper with current automation, scale up, automate new requirements, or abandon; possibly adopt other tools, transition out of GUI only, revisit pre-planning construct

# Maintenance and Sustainment Phase



- Manage automated software suite configuration
  - Configuration control essential; periodic verification/validation to account for changing system under test, update automation framework documentation; decide best tools, script access, and file mgt sys
- Update automation code
  - Respond to changing program/system, improve metrics like percent coverage, incorporate updates to automation software, consider other tools, update code documentation
- Manage scripts
  - Create repository of reusable scripts, expand to other automation efforts, search for new scripts and tools that can be modified for program testing needs
- Track program software defects
  - Implement disciplined FRACAS-like closed loop tracking system; identify POCs to own failure; conduct failure review boards
- Assess defect discovery trends
  - Statistical modeling, decide when to reduce testing frequency and depth, determine focus areas for future automation efforts, trends from root cause analyses