9-5-2017

Automated Software Testing in the DoD: Current Practices and Opportunities for Improvement

Darryl K. Ahner
Air Force Institute of Technology

James Wisnowski
Adsurgo LLC

James R. Simpson
JK Analytics

Recommended Citation

Ahner, D. K., Wisnowski, J., & Simpson, J. R. (2017). Automated software testing in the DoD: current practices and opportunities for improvement. Journal of Defense Analytics and Logistics, 1(1), 88–91. https://doi.org/10.1108/JDAL-09-2017-0017

Received 4 September 2017
Accepted 5 September 2017

Automated software testing in the DoD: current practices and opportunities for improvement

Darryl Ahner
Department of Operational Sciences, Air Force Institute of Technology, Wright Patterson AFB, Ohio, USA

James Wisnowski
Adsurgo LLC, Pensacola, Florida, USA, and

James R. Simpson
JK Analytics, Niceville, Florida, USA

The concept of automating the testing of software-intensive systems has been around for decades, but the practice of automating testing is scarce in many industries, especially in the government defense sector. A one-year project initiated by the Office of the Secretary of Defense (OSD), Scientific Test and Analysis Techniques Center of Excellence (STAT COE) and sponsored by Navy OPNAV N94 set out to:

- study the degree to which the Department of Defense (DoD) has adopted automated software testing (AST);
- share the best software practices used by industry; and
- develop and distribute an AST implementation guide intended for program management and novice DoD software test automators. The Current State of Automated Software Testing in the Department of Defense, AST Practices and Pitfalls Guide, and the AST Implementation Guide are available at www.afit.edu/stat/

The first document detailing the current state of AST in the Department focused on capturing the keys to successful DoD software test automation, collecting and sharing case studies from early test automation adopters and providing insights along with recommended strategies. The overarching finding was the difficult balance for leadership in dealing with the cultural and technical challenges to ensure that software test automation programs result in a positive return on investment (ROI). The focus on test automation ROI for this potentially disruptive technological program must consider budget impacts, manpower required and automation tool familiarization time, as well as both the net-test time savings and the associated increase in test coverage of capabilities of the software under test. Some valuable lessons learned were:

Journal of Defense Analytics and Logistics
Vol. 1 No. 1, 2017
pp. 88-91
Emerald Publishing Limited
2399-6439
DOI 10.1108/JDAL-09-2017-0017

- realizing the importance to not rely solely on the brittle record/playback AST tools;
- understanding the automation process needs to be treated as a software development effort;
- ensuring that the automation project is staffed with software developers familiar with Java, HTML, xml, etc.;
- considering the test scripts needed to be well-thought out in design and will require many iterations before production;
- accounting for the script and automation framework maintenance that needs to be planned and budgeted for ahead of time; and
- recognizing that regression testing and other repetitive tests offer the best candidates for high ROI from automation.

We focused on documenting themes, findings and automation activities common to all the services. Then each service is detailed including specific programs, organizations and service-unique practices. Specific areas addressed in the document are:

- policy and guidance;
- AST initiatives;
- culture for AST;
- notable programs applying AST;
- centers of excellence;
- AST tools experience; and
- AST metrics and coverage.

Although there is some overlap across these topics and within programs applying automation, these general areas can be used to tell the story of automation within and across services of the DoD.

The second document from the study, "AST Practices and Pitfalls Guide", details the current best practices along with challenges as industry adapts these game-changing test methods. Clearly, companies producing massive software solutions (such as Microsoft, Google and Apple) have been successfully automating much of their software testing, especially early in the development at the software unit level. The DoD faces the challenge that much of the early testing is done by the defense contractor, and by the time software-intensive systems are handed over to the military, the testing is primarily integration-, functional- and performance-driven and is primarily related to front-end, black box testing. The STAT COE spent about six months researching and interviewing acquisition program civil service and contractor professionals implementing AST or considering using automation for at least a portion of their testing.

The review of industry highlights the incentives driving AST – incentives that do not always apply to DoD concerns. However, the investments and struggles of AST in the industry provide a good vignette for the DoD community to observe and leverage its efforts, as it embarks on injecting more automation into software testing.

The third document, "AST Implementation Guide", focuses more on the operational and tactical level intended to serve those in the DoD interested in applying automation to software testing. It applies a systems engineering process based on the scientific method to the steps to conduct and achieve an automation capability along with the important need to perform a ROI analysis to make the business case for automation.

The document is organized around the phases of implementation listed below, which are intended to encompass the life cycle of automated software testing within an organization's test program. Each phase has specific tasks, metrics and deliverables that are explained in detail:

- *Pre-plan*: Research, invest time and gather information for making an informed decision on automation. Perform a cost-benefit analysis and compute an ROI; be sure to include any long-term benefits and then decide whether to automate. Specific steps include research into the program test plan, automation capabilities and opportunities, knowing man power skill set and resource needs and quantifying costs and benefits from automation.
- *Plan*: Develop an automated software test plan by identifying and prioritizing test requirements, identifying and assessing appropriate automation tools with quantifiable and discernable metrics, identifying barriers to automation implementation, drafting the test automation framework and outlining the test script needs.
- *Design*: Take the automation plan to 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 the configuration control is established, all culminating in a design review.
- *Execute*: The activities and decisions that enable 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, and then deciding the best automated test environment, integrating the tools within the designed test framework, developing and refining the automation scripts and iteratively testing out the execution while refining the process.
- *Analyze*: The focus is on the output of each automated test, typically involving recording data files or log files. The purpose is to combine, manipulate and analyze the output data to learn output errors and faults associated with the system under test (SUT) to include integration issues. Individual steps include setting the data format, assessing the output data, ensuring that anomalies are real and characterizing anomalies, revising automation metrics and ROI.
- *Maintain*: This final phase is often the most time-consuming and painful aspect of automation. Once test scripts have been written, executed and refined for optimal use, something (SUT, the test environment including monitors or operating system versions and IT patches and automation tool version) changes. The scripts now fail to execute properly unless revised, which is one of the tasks in the maintenance phase.

Although the phases can be visualized and enacted in a chronological or linear fashion, we realize and emphasize that there is significant connectivity between them such that moving in a less-structured or iterative direction can be advisable. The suggested approach also involves maturing several important automation tasks across multiple phases. For example, automation tool selection is often considered a primary and critical decision. The implementation guide suggests tool selection and tool acquisition to be a part of each of the plan, design and execute phases, where increased knowledge and topic maturity is obtained in subsequent phases. Iteration and looping of the phases are a key to success.

Feedback on these documents has been favorable as they have been briefed to the naval automated test and analysis executive board, at conferences (DoD/NASA Knowledge Exchange Conference, International Test and Evaluation Association [ITEA], to the Military Operations Research Society Symposium [MORSS]) and to the stakeholder organizations across DoD. The overall project is entering a second phase in FY18 that will enhance these deliverables, create a repository for AST tools and scripts and continue direct program support to some Navy organizations at various stages in their automated software testing journeys. The results of this effort enable the acquisition community to inject more rigor during the development phase that also enables a capability that supports demonstrating assurance during operational acceptance and follow-on system improvement efforts.

Corresponding author
Darryl Ahner can be contacted at: darryl.ahner@afit.edu