Efficiencies of Virtualization in Test and Evaluation

Elfriede Dustin, IDT
Tim Schauer, IDT

Abstract. Using automated testing in a virtual test environment can reduce the time and effort required to complete test execution and data analysis, significantly reduce test suite costs, and at the same time increase the thoroughness of system testing.

Section 1: Introduction

NIST produced a report in 2002 titled, "The Economic Impacts of Inadequate Infrastructure for Software Testing."[1] This report "estimates the economic costs of faulty software in the U.S. to range in the tens of billions of dollars per year and have been estimated to represent approximately just less than 1% of the nation's gross domestic product." The report goes on to state that "based on the software developer and user surveys, the national annual costs of an inadequate infrastructure for software testing is estimated to range from $22.2 to $59.5 billion."

Also in 2004 the Chief of Naval Operations (CNO) Guidance included direction to the Commander, Operational Test and Evaluation Force (COMOPTEVFOR) to lead a collaborative effort among Navy, OSD, and contractors to reduce the costs of Test & Evaluation (T&E) by 20%. In developing a response to the CNO Guidance for 2004, COMOPTEVFOR surveyed programs and included the following as T&E cost drivers:

- **Redundant testing**
- **Significantly increased levels of regression testing driven by technology insertion**
- **Increasing complexity of computer software testing, to include systems of systems**
- **Interoperability testing and certification**

Based on the COMOPTEVFOR findings, more effective approaches for testing are needed to be able to meet the CNO Guidance to reduce T&E by 20%.

A GAO Report to the Congressional Committees dated June 2012, describes that "recent defense acquisitions have experienced from 30% to 100% growth in software code over time."[2]

With the increased size and complexity of systems of systems testing, requirements for unique / duplicate test facilities and test-beds for major Navy product areas, software testing is rapidly becoming the "very longest and most expensive pole in the tent" when it comes to fielding new capabilities. Because of many reasons including organizational boundaries, lagging technologies, unique requirements, and testing methodologies, the testing of new capabilities being fielded has become a significant cost and time element of the process and without some form of change to the current process, could become even more significant. Some estimates have it consuming more than 60% of the time and cost of the process.[3]

Our experience at IDT shows that using virtual test environments with automated testing using Automated Test and Re-Test (ATRT) can help reduce testing infrastructure cost for testing areas such as interoperability, system testing, functional testing, component and unit testing. Additional benefits of automated testing in a virtualized environment include a more reliable system, improved testing quality, and reduced test effort and schedule. A more reliable system results from improved performance testing, improved load/stress testing, and improved system development life cycle through automated testing. The quality of the test effort is improved through better regression testing, build verification testing, multi-platform compatibility tests, and easier ability to reproduce software problems. Test procedure development, test execution, test result analysis, documentation and status of problems are also activities benefiting from automated testing.

ATRT in a virtual test environment can provide a stable, scalable, affordable and accessible automated testing infrastructure that extends across one or many server farms, across one or many System(s) Under Test (SUTs) and works with a common set of cloud computing concepts to support a broad virtualized enterprise automated test environment. This specific testing setup allows the use of virtualization in a specialized way to reduce the need for purchasing, storing and maintaining various expensive test environment hardware and software. Proper virtualization setup provides a multi-user access automated testing solution that allows users to implement and reuse ATRT, along with all testing artifacts, on a provisioning basis. Additionally all related automated testing activities and processes, i.e. test case and requirements import; requirements traceability, automated test creation and execution, and defect tracking take place in this virtualized environment.

Combining ATRT test efficiency with the hardware cost savings implementing in a virtualized/cloud environment, the resulting estimated savings are tremendous. For example 20 Virtual Machines (VMs) fit on 1 server in our virtual environment example – allowing for huge savings in the test environment, i.e. in this case a 20 to 1 cost savings. Additionally, in the virtual environment, the SUT VM can be located anywhere on a connected network and does not need to be located physically in the same VM as the testing VM.

Examples of automated software testing in a virtualized test environment include:

1. **Automatic provisioning of a virtualized automated test environment**
2. **Automatic provisioning of the entire automated testing lifecycle for any type of SUTs**
3. **Continuous integration using virtualized environments**

Sections 2.0 through 4.0 provide technical overviews of the various embodiments of the present ATRT/Virtual Test Environment (VTE) implementation.

Section 2: Automatic Provisioning of a Virtualized Automated Test Environment

As shown in Figure 1, the virtualized setup allows for a stable, scalable automated testing infrastructure that extends to one or many SUTs or one to many automated testing tool installations (in this example ATRT). This virtualized test environment setup is a highly scalable solution whether a user needs to run 10 or 10000s of tests connecting to N number of SUT displays and servers over days or weeks and whether the user needs to analyze 100s of test outcomes or 10000s or more.

In order to support a virtualized automated test environment, it is critical the automated testing solution itself be scalable. For example, the ATRT technology allows for N number of concurrent tests to run or N number of serial tests, depending on the test type required. All of the tests and test outcomes are stored in the ATRT database/repository for access by any subscriber (or user) of the ATRT virtualized environment. A subscriber/user can be a developer or tester or anyone on the program with ATRT user access privileges.

This virtualized test environment example setup supports live migration of machines; load balancing; easy movement of machines to different servers without network interruption and allows any upgraded VM to run on any server. As a result, it is also important that in a virtualized environment an automated testing solution is not only scalable but portable. ATRT can test systems independent of OS or platform so it is able to support applications running on both Windows and Linux providing flexibility to migrate machines without the constraint of the OS the automated test solution can support.

Additionally, an automated testing tool should be selected that does not need to be installed on the SUT. ATRT is an example of a solution that does not need to be installed on the SUT and instead is communicating with the SUT via a VNC Server or the RDP protocol which transmits the SUT images back to the tester to the ATRT client. Few tools exist that do not need to be installed on the SUT. The typical automated testing tool needs to be installed on the SUT so it can link to the GUI coding libraries to get the object properties of the GUI widgets and/or pull information out of the Operating System's window manager in order to create an automated test baseline. Installing an automated testing tool on the SUT however is generally not desired, because 1) the installation modifies the system environment (the testing system environment should be identical to the production system environment) and 2) it does not lend itself to cloud computing because of the additional tool installation on each SUT.

In this VTE the SUT VM can be located anywhere on a connected network and does not need to be located physically in the same VM as the ATRT VM. This allows for tremendous flexibility, for example multiple ATRT VMs can run in the VTE connecting to 100s of SUT VMs. However, in the typical automated testing setup where the tool needs to be installed on the same machine as the SUT, a 1 : 1 setup is required, i.e. 1 Automated Testing tool for each 1 SUT, negating some of the savings expected in a VTE.

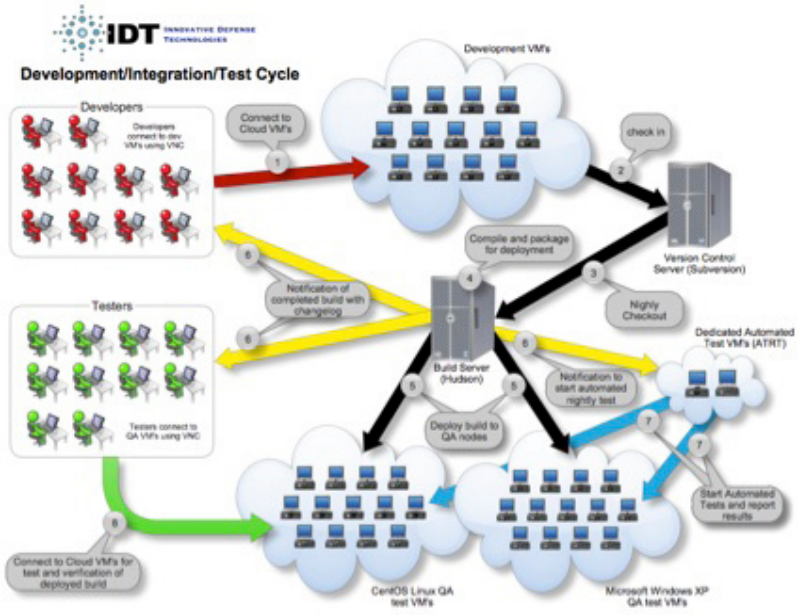

Figure # 1: Top-Level Block Diagram of the Automated Provisioning of the ATRT virtualized Test Environment

Section 3: Automatic Provisioning of the Entire Automated Testing Lifecycle for any Type of SUTs

One or many users can access a VTE one at a time or concurrently with any device such as a laptop, iPad, iPhone, etc. with nothing installed on their device but a network connection enabling the capability to login to an IP address to connect to the ATRT virtual environment.

Users can then request one or more instances of a VM along with the automated testing tool. The automated provisioning meets a user's changing needs without the users being required to make any software modification on their end as required to conduct the automated test. The VTE in this example can spawn an instance of ATRT which then allows the user to access any automated testing artifact and execute the automated testing lifecycle. The user can then conduct any activity that is part of the automated testing lifecycle, i.e. create an automated test case, reuse or troubleshoot an existing automated test case created by any user, import requirements, produce a requirements traceability report. The VTE provides any additional features and capabilities required to support the SQA process and help improve Quality, such as Unit Testing and Code Coverage.

Exemplary features of this process include:

- **Developers update the code on the development VMs**
- **Developers check in their code into Version Control**
- **Build Server conducts automated nightly checkouts**
- **Build Server compiles and packages a new build for deployment**
- **Nightly automated tests are run**
- **Users are notified of the automated test outcome**
- **Build Server deploys the new build to the QA nodes**
- **Testers access the QA nodes and create and/or run their automated tests**
- **Testers, Developers, and all users conduct the automated testing lifecycle activities and maintain all ATRT test artifacts in the virtual environment**

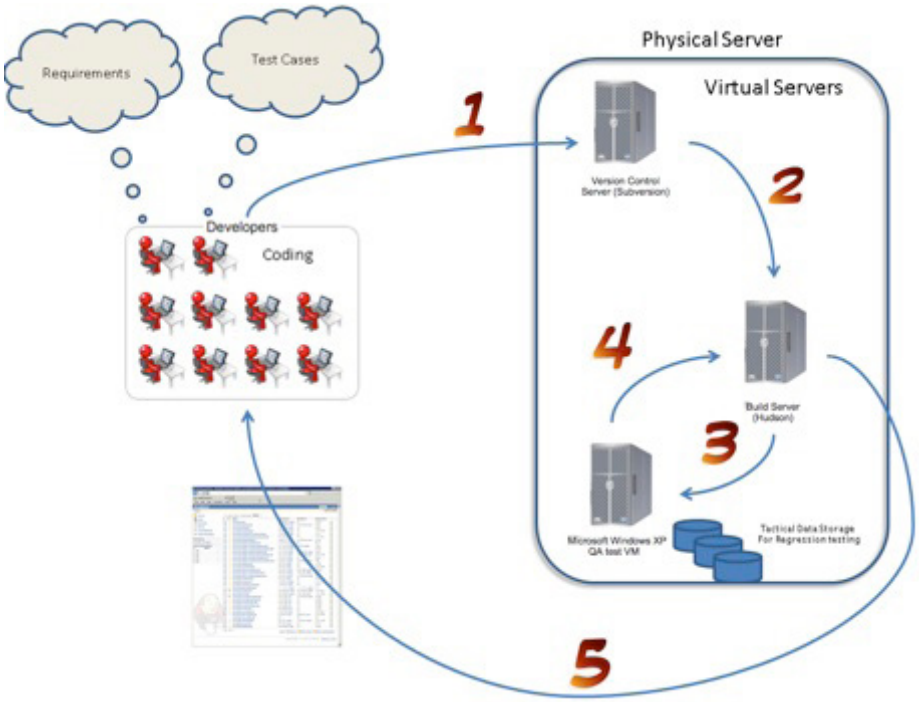

Figure 2. Continuous Integration Environment Example

Using the virtualized test environment a single engineer may control an entire test of complex systems with only his/her iPad, laptop, etc. and only requires access to the network.

Section 4: Continuous Integration Using Virtualized Environments

Continuous integration is an industry adapted software engineering best practice in which any change to the code or environment is tested and reported on as soon as feasible. In most cases this involves nightly software builds and nightly automated test runs to allow for quick look reporting on any newly introduced issues. Virtualized test environments play a major role in this best practice.

The development environment that makes this possible is one of a virtualized environment combined with both regular workstations and laptop computers networked together.

1. Developers first review the system level requirements and create a set of automated tests. Code is locally edited / compiled/linked and then checked in to a virtualized version control repository, such as SVN. From here other developers can check out both updated code off of the trunk or from code from specific branches to support different build.

2. Upon code checkin, a continuous build server, such as the Hudson Continuous Build virtual server is triggered to start a complete build/check/test/report cycle. Hudson will perform the following tasks:

a. Update the latest code from SVN

b. Compile the code and check for compile errors

c. Link the code, check for any link errors

d. Perform source code style checks and copyright checks

e. Start a series of both internal and external regression tests:

i. Internal regression tests will execute automated tests to verify key use case tests to verify results are as expected and also ensure that code that was updated has not adversely affected the existing functionality.

ii. External regression testing can then utilize any automated testing capability on another virtualized node to perform tests as an end user would be expected to do (i.e. through a GUI interface). Each test can then analyze hundreds of system level requirements. Each requirement may itself be verified hundreds to thousands of times. External regression testing again compares its results against a known good set of results.

3. The internal and external testing results are then reported back to the Hudson server. Upon completion of successful internal and external regression testing, the Hudson server continues to now build an installer package that will be available to the end user at fielded locations. Additionally, key statistics are gathered on the entire process and saved for later retrieval.

4. Finally, Hudson provides the developer with reports on the entire sequence of testing. The developer can then use the results of the testing to make appropriate code changes.

Section 5: Summary

Using automated testing in a virtual test environment we have been able to demonstrate the ability to reduce the time and effort required to complete test execution and data analysis, significantly reduce test suite costs, and at the same time increase the thoroughness of system testing. An increase in software testing thoroughness equates to a reduction of defects found in the field and reduced total ownership cost. Automated testing in a virtualized test environment will also enable much earlier identification of integration and interoperability characteristics of any software products that must interact with other systems. Identification of software specific integration characteristics in products in-stride with software development cycles enables the identification of issues to also be decoupled from the delivery of the final hardware configuration.

NOTES

1. See <http://www.nist.gov/director/planning/upload/report02-3.pdf>
2. <http://gao.gov/assets/600/591608.pdf>
3. Hailpern and Santhanam, 2002 (The cost of providing [the assurance that a software program will perform satisfactorily in terms of its functional and nonfunctional specifications within the expected deployment environments] via appropriate debugging, testing, and verification activities can easily range from 50 to 75 percent of the total development).

ABOUT THE AUTHORS

Elfriede Dustin is Director of Solutions at IDT where she works on developing new ideas and discovering new approaches to the requirements based automated software testing challenge. Software development is still an art and that makes automated software testing a special challenge. IDT (www.idtus.com) strives to meet that challenge by producing a reusable automated testing framework that includes reusable automated testing components, starting with requirements through the entire software testing lifecycle to defect closure. Elfriede has a B.S. in Computer Science with over 20 years of IT experience, implementing effective testing strategies, both on Government and commercial programs. She has implemented automated testing methodologies and testing strategies as an Internal SQA Consultant at Symantec, worked as an Asst. Director for Integrated Testing at the IRS Modernization Efforts, implemented testing strategies and built test teams as a QA Director for BNA Software, and was the QA Manager for the Coast Guard MOISE program.

She is the author and co-author of 6 books related to Software Testing, i.e. author of the book "Effective Software Testing" and lead author of "Automated Software Testing" and "Quality Web Systems," and co-authored the book "The Art of Software Security Testing," together with Chris Wysopal, Lucas Nelson, Dino D'ai Zovi, which was published by Symantec Press, Nov 2006.

Together with IDT CEO Bernie Gauf and IDT FSO and Sys Admin Guru Thom Garrett she wrote her latest book "Implementing Automated Software Testing."

E-mail: edustin@idtus.com

Tim Schauer graduated from the University of Wisconsin-Madison in 1985 with a Bacholor of Science degree in Physics and a B.S. in Astro-physics. He received his commission in the US Navy and worked as both the weapons officer and communications officer on the USS Shenandoah. After the Navy, Mr. Schauer worked on Tactical Software for the SPY-1A Phased Array radar at the Naval Surface Weapons Center in Dahlgren, VA. He then became testing lead and lab manager for the SeaWolf Class / BSY-2 integration facility in Moorestown, NJ. Later, he worked as senior logistics analyst for US Pacific Command at Camp Smith, Hawaii.

Tim Schauer has been working with Virtual Servers since first being introduced to them at Pacific Command (PACOM) in the late 1990's. He continued to develop virtual systems while working at the San Diego Data Center for the County of San Diego and Children's Hospital of Los Angeles. He has virtualized over 90% of the Beaufort County, South Carolina, library's IT system, greatly reducing cost while increasing productivity. Finally, Tim is currently working on virtualizing a US Navy Tactical Weapons System to facilitate ongoing ATRT automated testing at the IDT facilities in Arlington, VA.

E-mail: tschauer@idtus.com
Phone: 843-473-5465