SCTE CABLE-TEC EXPO '13
OCTOBER 21-24 / ATLANTA, GA

SCTE Society of Cable Telecommunications Engineers

Automating Testing to Meet Enterprise SLAs

A Technical Paper prepared for the Society of Cable Telecommunications Engineers
By

Alex Henthorn-Iwane
Vice-President, Marketing
QualiSystems
2880 Lakeside Drive
Santa Clara, CA 95054
+1-408-588-1260
alex.h@qualisystems.com

Overview

Network quality affects business. And where quality issues arise is important[1]. Finding problems in the field, versus finding problems during testing can make the difference between customer satisfaction and Service Level Agreement (SLA) payouts. This makes network testing a critical function, and the effectiveness of network testing a worthy topic for business analysis. Unfortunately, today testing tends to be both very costly and quite inefficient due to highly manual, un-automated processes and outmoded architectural approaches. To paraphrase Tom Demarco's well-known line from his seminal book on managing software teams, Peopleware, "While the machines have changed enormously, the business of [network testing] has been rather static".

If there isn't enough reason to evolve for cost savings, time to market, customer satisfaction and competitiveness reasons, looming technology sea changes should add a good deal more motivation. Software defined networks (SDNs) and network function virtualization (NFV) are challenging long-held assumptions about network architecture. They also promise to make networks more agile and will definitely make them more complex. The sheer change that these technologies can bring in how networks are built, plus the fact that they are purpose-designed to enable much higher levels of dynamism in operating networks and delivering services make it imperative to ensure that network testing has the agility to ensure quality service delivery.

In short, the time is now to update network testing practices. To do so requires modernizing two inter-related domains: test lab operations and test automation. Organizations that make sound investments in automation will reap substantial rewards in the form of significant capital expenditures (CAPEX) and operations expenditures (OPEX) savings in their testing organization and infrastructure, accelerated time to market with new services, and higher service quality that builds customer satisfaction, business relationships and brand power. This paper delves into the state of test lab infrastructure management and test automation today, and then explores the state of the art in both arenas.

Content

The State of Test Lab Infrastructure Processes

Test labs are typically established to serve as a shared, dynamic infrastructure for development, Quality Assurance (QA), technical support and field engineers to perform a variety of critical testing tasks. These test labs centers typically include many instances of multi-vendor equipment representing the full stack of computing, storage, network, and virtualization components that are used to deliver services. Two major categories of test activities tend to dominate these test lab environments:

- Interoperability or certification tests to ensure that new products will work properly in combination with a variety of other products that are installed in target deployment environments. For service providers and cable MSO's, this may include certifying new devices and software against all relevant production architectures, testing against common customer deployment architectures, and custom testing for strategic accounts.
- Technical issue replication and fix verification. In this use case, when a field customer issue is reported, technical support engineers and must assemble the same scenario and configuration in which the field problem occurred and replicate it so that escalation engineers can examine the problem in context. Once a fix is received (typically from a vendor), support engineers must then verify that the problematic behavior is resolved in that same configuration before deploying the fix to the network.

Both of these testing use cases require the dynamic reuse of multi-vendor equipment and virtual resources in variable configurations or test topologies. For example, a test may require verifying the interoperability and performance of a new version of a cable modem termination system (CMTS) against a test topology consisting of Ethernet switches, Internet Protocol/Multi-Protocol Layer Switching (IP/MPLS) routers, head-end servers, home gateway devices and end-user equipment. The simplicity and

speed with which infrastructure can be deployed to support these tests is an obvious and important factor.

Unfortunately, in the vast majority of test labs today, the process to deploy costly resources into test topologies is anything but simple and fast. Ironically, the networking, storage and computing technology that is being tested is increasingly virtualized, agile and software-controlled. Yet, the typical test lab's infrastructure management setup provisioning processes are resoundingly manual in nature. This can be seen in a number of ways:

- Absence of inventory visibility. In most test labs, equipment inventory is not tracked in a way that provides visibility to engineers. While most organizations perform asset tracking for financial purposes, what passes for the inventory management used by engineers is a spreadsheet that is often ill-maintained. As a result, it is difficult to tell without exhaustive work what equipment exists, is being used by whom and what is truly available.
- Offline test topology design. Since there is no usable inventory visibility, it follows that test topology design is done completely offline without regard for resource availability. Visio or other diagramming tools are most common, and basically produce the electronic version of a paper drawing, which is usually then printed to aid in a time-consuming manual hunt for relevant equipment.
- Chaotic connectivity management and costly errors. Once inventory is found that is at least apparently available, engineers must manually re-cable connections between the equipment. With multiple people making adds, moves and changes, typically without up to date documentation, errors such as disconnecting someone else's test inevitably occur. Test breaks are a sadly common reality in most test labs today.
- Manual provisioning. Once an engineer has painstakingly assembled a physical topology, (s)he must then perform a variety of further time-consuming device and logical provisioning steps. For example: loading a particular hyper-visor version, changing operating system (OS) images on networking equipment, setting up logical connectivity between servers and virtualized storage devices, and instantiating virtual machines. Test engineers may be highly educated on the components they are testing, but they spend the vast majority of their time on low level provisioning tasks.

The result of these manual setup processes can be encapsulated in one word: inefficiency. The waste of space, equipment and power is evident through a number of indicators:

- Large ratio of test setup to actual testing. Test engineers can easily spend a week in the setup process for a test that takes less than a day to run.
- Hoarding and poor resource sharing. Given the time that engineers often have to put into physically locating, connecting and provisioning test resources into a ready-to-use test topology, it is understandable that they don't want others to cause any changes. This time-consuming setup process also makes it literally too costly to release resources if a test isn't immediately ready to start, meaning that costly networking or other components may sit idle, most often powered up and consuming power, for days or even weeks until the test is ready to run.
- Very low device utilization. Tens of millions of dollars in test lab capital equipment are typically only 15% to 20% utilized.
- Excessive power usage. The low percentage of time that devices are productively utilized belies the fact that most devices are still left powered on nearly 100% of the time, which adds up to a lot of wasteful power consumption. Test lab managers have reported experiences where proactive power downs of large test environments have resulted in only a few test engineers complaining, leading them to wonder why all the other equipment was powered on continuously in the first place.

There are significant implications to the level of waste created by manual operating processes in test labs. To begin with, a device utilization of under 20% means that as demands for testing grow with the deployment of new services, devices and software, the pace of investment in test lab capacity will rise at a rapid rate. With test labs costing anywhere from $1K to $3K per square foot inclusive of equipment costs, this can lead to huge, unnecessary CAPEX outlays over time. Wasted power costs due to so much equipment not being utilized productively yet still powered up are also daunting. Assuming a modest $50.00 annual power cost per square foot for a typical 50 KW/square foot data center facility[3], the expansion of capacity needed to accommodate low device utilization can lead to hefty additional OPEX.

Above and beyond the bottom-line of wasted test data center capacity due to manual processes, there is also the top-line issue of slower business velocity. When setup to testing ratios are as high as 80:20, this means that testing cycles are longer, which either delays service releases or causes organizations to compromise test coverage and quality which leads to higher incidences of problems found in the field that are much costlier to fix

after products are released. Lower network quality leads to higher incidences of SLA payouts, damage to customer satisfaction and key account relationships, and lower overall competitiveness.

The State of Test Automation Processes

Test lab infrastructure management is not the only aspect of the testing process that commonly suffers from inefficiencies. Many network tests today are performed manually, rather than in a software automated fashion. Even in cases where automation is applied, automation projects themselves often experience difficulties in delivering on their promises because they are based on a script-based architecture. Scripting is a necessary and useful programming method for creating automation. However, "script-based architectures", meaning collections of lengthy script documents, suffer from a number of significant drawbacks that impact efficiency:

- **Dependence on programmers and high maintenance costs** – scripts are typically coded to automate a multistep process, generating lengthy files of TCL, Perl, Python or other scripting language. These files are lengthy, complex, and thus difficult to maintain, update or repurpose as conditions in the testing environment change. Since only programmers create or update scripts, the entire automation process becomes very tied to individual programmer knowledge. If there is a change that makes scripts out of date, programmers must then spend significant amounts of time to create new version of the scripts. In many cases, scripts are written without any comments, so if the original script developer is no longer available, it may be impossible for new engineers to adapt or update the script and force them to write it again from scratch. This process is costly in terms of programmer time and becomes a bottleneck in the productivity of the majority of non-programmer testing personnel.

- **No scalability, low penetration** – due to the high cost of maintenance and because most testing environments experience regular changes that demand continuous and time-consuming script revisions, scalability is very difficult to achieve. As a result, many automation projects never go above 10% penetration of testing processes.

- **Script bloat** – over time, a large number (in fact, the majority of generated scripts) cover only a minority of testing functions. This core of automated tests exists in a multitude of versions that cover minor variations in the ever-changing test environment. The proliferation of scripts becomes a logistical and revision-control problem, which

requires the introduction of source version control (SVC) tools. This further burdens programming staff with additional administrative overhead.

- **Vulnerability to disruptions due to changes in personnel** – since the content of the scripts is only known to and maintainable by the programming staff and often only by the actual developer, there is very little systematized knowledge. If programmers change their job or role, their expertise is lost and the viability of the automation project may be threatened.

- **High total cost of ownership (TCO), low return on investment (ROI), and project fatigue** – programmers are specialized staff, and therefore expensive. Never-ending, heavy programming requirements make the TCO of traditional automation projects unacceptably high. Project costs become more glaring since due to the low penetration of automation into the body of testing tasks, the overall testing process does not accelerate significantly. This means that the return on investment is poor. Lacking the penetration to reach positive ROI and burdened with a high TCO, automation projects risk losing funding and sustainability.

Automating the Test Lab

The state of the art for test lab automation has advanced dramatically. Modern lab management automation software should deliver a broad range of capabilities that enable the following capabilities:

- Centralized, live infrastructure and resource inventory that is customizable to make it easily searchable
- Inventory-aware test topology design
- Shared, calendar-based resource and topology reservation
- Connectivity mapping and automated connectivity control
- Easy to create automated provisioning tasks
- Non-programmer friendly automation workflow creation based on a library of highly reusable test objects that can be created from a wide variety of sources and leveraged to create:
    - Auto-discovery, auto base-lining and other automated maintenance routines
    - Full test automation workflows

Another important concept for sustainable automation is that the platform for managing the test lab should avoid the pitfall of using monolithic and fragile script-

based approaches to automation, which cannot scale due to their high maintenance costs. An object-oriented platform that captures and manages all inventory resources, test topologies, provisioning actions and testing tasks in a library of highly re-usable, easy to update object building blocks is the only architecture that ensures automation scalability and long-term and cost-effective sustainability of Lab as a Service (LaaS) cloud administration.

Best Practices for Test Lab Infrastructure Management

To achieve a successful test lab infrastructure automation roll-out, best in breed technology is critical but must be accompanied by best practice methodologies:

Highly Automated Physical Layer Connectivity:
Software-based automation benefits from a structured, documented, and easy to operate physical connectivity environment. The state of the art practices in LaaS consolidation include deployment of Layer 1 switching to virtually eliminate manual cable patching. Of course, Layer 1 switching should be combined with sound, TIA and SCTE standards-compliant data center, headend, and hub layout and structured cabling[2] so that the entire physical environment can flex to changing requirements over time.

Resourcing the Automation Infrastructure Service
The most successful lab automation deployments involve dedicating personnel resources with data architecture and programming skills to build and maintain the object library of inventory resources, test topologies, provisioning and shared testing objects and workflows. The broader user community can then leverage this library to build and reserve topologies, easily perform provisioning, and progress into test automation as the library is built out. Dedicating resources to maintaining the object library as an infrastructure service is strongly recommended, since if the utility and ease of use of the object library is not maintained at high levels, users will abandon the automation system, wasting the investment.

A Phased Approach:
Successful lab automation is typically built in phases, where each phase aims for a visible productivity gain and return on investment in a reasonably short time in order to build user engagement and momentum and create realistic expectations. Generally speaking, achieving "hands-off" visibility and reservation of lab resources using Layer 1 switching and automation software is the first major goal. This level of automation allows remote users to be on an equal productivity footing with local users, and promotes deep buy-in of all user groups with the consolidation

initiatives. Any LaaS consolidation project should ensure that the centralized data center is designed with this in mind.

The second automation phase is to free testers from the time-consuming tyranny of low-level device provisioning tasks. This involves turning manual provisioning processes into easy to invoke, menu-driven tasks from the automation graphical user interface (GUI). The best practice in this stage is to ensure the sustainability of the system, by avoiding reliance on fixed scripts. While it may be relatively easy to create a first set of provisioning scripts for some usage scenarios, the time-consuming nature of script maintenance will too often cause the provisioning capabilities of the system to become out of date. The negative experience of using scripts that don't work will end up alienating users and deepen their reliance on manual processes. Automation of provisioning tasks typically start with the basic provisioning steps needed to get the devices under test (DUTs) to a particular state, such as uploading OS images or applying patches. More advanced provisioning tasks involve common configuration steps to ready the logical layer of a test topology, such as configuring virtual local area networks (VLANs), routing adjacencies, or tunnels on physical or virtual switches. These automated provisioning objects help test engineers more easily accomplish the routine tasks that often dominate their workdays, and allows them to focus more on higher order thinking to achieve maximal test coverage.

A third phase of test lab infrastructure automation that is short of full test automation is to create automated maintenance routines. Examples include auto-discovery, which helps keep the inventory up to date, and auto-base lining returns devices back to their default provisioning states on a timed basis. These types of routines require development of a comprehensive set of device control/interface automation objects for all necessary devices in the test infrastructure, so that they can be leveraged across multiple maintenance automation processes.

Tiered Resource Domain Access:

In a large, shared LaaS cloud, not all users are created equal, which means that there needs to be a way to offer varied tiers of access to lab resources to different types fo users such as:

- Global administrators who should have access to all resources
- Domain administrators who have purview over a sub-set or domain of the lab
- Power users—who are given visibility to one or all domains, and who have both visibility, topology design and resource reservation rights
- External users—such as contractors or third-party testing organizations who may have visibility to a domain of resources, but must request resource reservations rather than

- Self-service users— such as sales engineers or even customers, who are only offered a pre-set menu of reservable topology resources for pre-defined purposes such as technology demonstrations or proof of concept tests

The Path to Testing Velocity—Object-Oriented Test Automation

A modern test lab automation software platform will go beyond hands-off test topology design and automated provisioning. It will also provide the way to implement full test process automation. Like lab infrastructure management, test automation has advanced considerably compared to traditional script-based architecture approaches. Testing organizations can gain a large improvement relative to the high TCO and low ROI of traditional test automation by implementing an object-oriented model. Instead of creating long, monolithic, hard to maintain scripts, an object-oriented approach enables the capture of all automation elements as building block objects. This includes objects for interfacing with test lab infrastructure resources (compute, storage, network, virtual, cloud), provisioning actions (such as loading OS images), and testing tasks (such as running a traffic load test). An object-oriented architecture offers a quantum leap in maintainability compared to scripts:

- The limited scope of automation objects means that they are easy to capture, maintain, and refactor to meet the requirements of a changing test environment
- A shared library of resource, provisioning and testing objects can be maintained in a systematic fashion. While programmers and data architects are the ideal personnel to build the library, the easy maintainability of the object library reduces risk because there is a greater balance between the expertise residing in the system vs. that in programmers' brains.
- Automation objects can be tagged with arbitrary labels so that they can be easily searched and leveraged by many users from a shared library.

The greatest results come from combining a highly reusable object library with powerful GUI tools. Together, they allow for much more productive and efficient automation processes and practices:

- **Automation driven team wide** – non-programmers can easily use the object library and powerful GUI tools to drive all day-to-day automation processes. Key automation GUI tool capabilities include:
    - Drag-and-drop test topology design using physical and virtual test infrastructure resource objects

  - Right-click menu-driven provisioning actions from the test topology visualization GUI that leverages provisioning objects
  - Drag-and-drop automation workflow design using a library of user-generated testing objects, and out-of-the-box automation logic objects

  Of course, better results can be had if test engineers are offered training in software development principles, since this will promote the development of the most elegant, well-tagged and scalable automation code and objects.
- **High levels of reuse** – not only is the object library highly reusable, but generated test topologies, provisioning and test workflows can be saved and shared across teams, promoting a higher level of reuse
- **'Object transfer' vs. knowledge transfer** – going beyond the simple notion of 'sharing' between peers in the same department, an object library approach means that test topologies and workflows provide a highly accurate and efficient method of handing off precise scenarios between developers, architects, QA teams, operations, technical support, field personnel and even customers. This is a huge time saver, as knowledge transfer processes based on verbal descriptions, text write-ups, and static diagrams are time consuming and error prone.
- **Automation of results analysis** – one of the most time-consuming aspects of testing is sorting through the results. An object-oriented approach can ensure that test results are ordered and recorded in the most digestible format to feed reporting systems so that analysis reports are quick and easy to produce.

These vastly improved processes revolutionize test automation leading to lower TCO and higher ROI:

- Programmer time is maximized, restraining ongoing costs and lowering TCO
- Automation of testing tasks accelerates and achieves very high degrees of penetration – 80% to 90% of tests can be automated
- Testing cycles accelerate speed and expand coverage, leading to a strong ROI, faster time to market and higher quality
- Costly test infrastructure resources are optimally used through improved sharing, leading to huge CAPEX and OPEX savings

A Global 500 Telecommunications Telephony Case Study

Customer Profile:

A Global 500 telecommunications service provider operates worldwide, with revenues in the tens of $billions, and tens of thousands of employees. Service offerings include:

- Local and long distance voice telephony services
- Broadband Internet services
- Enterprise-class business data services
- Internet Protocol Television (IPTV)
- IP Telephony/Voice over IP (VoIP)
- Outsourced Information Technology (IT) and managed network services

Business Challenge:

The telecom operator provides managed IP and analog telephony services and equipment to enterprise customers, who expect flawless performance of their voice services. The managed telephony service must support a variety of different phones, such as VoIP desk phones and mobile IP phones. As new feature versions are released, this matrix of supported equipment must be extensively tested for a variety of use cases, including new phone deployments.

The engineering team responsible for service deployments faced competing pressures to release new feature sets and new phone support to customers in a timely fashion, but at the same time, to ensure the highest quality and prevent costly downtime or feature malfunctions. A significant challenge was that pre-deployment testing was very time-consuming and costly, requiring hiring multiple external contractors due to the heavy task load when performing regression testing for new feature releases or major new deployments.

The engineering team needed a test automation solution that could meet the following requirements:

- Ability to handle commercial voice test tools such as those provided by Ixia Communications, Spirent Communications, Empirix, GL Communications, and others, as well as open-source tools such as Wireshark/Tshark packet capture tools
- Ability to test GUI interactions such as the registration of VoIP phones on a web-based provisioning application
- End-to-end call testing

Test and lab Automation Software Applied

The telecom provider implemented a test and lab automation software solution to address lab management, device provisioning and test automation capabilities in order to optimize the entire testing lifecycle, including:

- Managing lab inventory including physical DUT and testing equipment, L1 switches, and virtual resources such virtual machines and switches in a live, searchable database tagged with testing attributes, eliminating manual searching for equipment in racks
- Test topology design via a drag and drop GUI environment that matches available inventory
- Calendar-based resource and test topology scheduling and automated device provisioning that dramatically cuts down on test setup time and increases costly device utilization
- Test automation including the ability to leverage existing automation scripts and integrate with management applications, as well as the ability to offer automation functions in an easy to use GUI workflow design environment so that non-programmers can boost their test automation efficiency.

Engineers were able to fully meet their technical requirements and have been able to automate test workflows such as the following:

- Phone provisioning testing
  - Register the phone with a web-based provisioning application GUI
- Setup and initiate packet capture tool
- Call flow testing
  - Configure and launch test to start a Session Initiation Protocol (SIP) call and answer
  - Configure and launch test to start an analog call
  - Test tools answer call, capture digits, analyzes digits and reports
- Stop the packet capture
- Analysis: Perform deep packet inspection analysis of packet capture traces
  - Pass/fail criteria for tests/regressions based on message sequencing and timing thresholds
- Phone de-provisioning test
  - Unregister the phone

Business Value

After adopting the test and lab automation solution, the engineering team has been able to achieve 80% automation in critical test areas, leading to the following benefits:

- Reduced dependence on external test contractors, resulting in significant cost savings
- Acceleration of testing processes through much higher efficiency, resulting in the capacity to consider adding another service feature release annually which makes the telephony service more competitive and satisfying to customers

In addition, test lab infrastructure management software dramatically increased efficient utilization of costly capital equipment in test labs, leading to significant savings since new capital purchases could be deferred.

The engineering team calculated that based on cost savings alone, a positive return on investment (ROI) on the investment was realized within 12 months of deployment.

Conclusion

Testing organizations have historically suffered from a lack of attention. With the growing complexity of networks driving ever higher levels of investment in test lab infrastructure and personnel, cable operators can no longer afford to neglect automation. Wise investments in automation test lab infrastructure management and testing processes will yield improved customer satisfaction, top line performance and bottom-line efficiency and profitability. For cable operators seeking to meet the increasingly stringent requirements of enterprise customers, automating testing can ensure SLA achievement and successful long-term positioning within the enterprise telecom services market.

References

Understanding and Controlling Software Costs

1. Understanding and Controlling Software Costs, B.W. Boehm and P.N. Papaccio, IEEE Transactions on Software Engienering, Volume 14 Issue 10, October 1988
2. TIA-942, Telecommunications Infrastructure Standard for Data Centers
3. http://slashdot.org/topic/datacenter/why-you-should-build-datacenters-in-the-u-s-power-costs/

Abbreviations and Acronyms

CAPEX	Capital Expense
CMTS	Cable Modem Termination System
DUT	Device Under Test
GUI	Graphical User Interface
IP/MPLS	Internet Protocol/Multi-Protocol Label Switching
LaaS	Lab as a Service
NFV	Network Function Virtualization
OPEX	Operating Expense
OS	Operating System
QA	Quality Assurance
ROI	Return on Investment
SDN	Software Defined Network(ing)
SVC	Source Version Control
Tcl	Tool Command Language
TCO	Total Cost of Ownership
TIA	Telecommunications Industry Association
VLAN	Virtual Local Area Network