Efficient Configuration Test Automation Using Virtual Machines

Vladimir Belorustes, Ph.D.

Plaxo, Inc.

***Vladimir Belorusets** has more than two decades of extensive experience in software development, test automation, test management, and software engineering methodology. He earned his Ph.D. in control systems from Moscow Institute for Systems Analysis, Russian Academy of Sciences. Vladimir held different positions from Senior Software Engineer to Director of Software Quality Assurance in various Silicon Valley high-tech companies. He also taught numerous courses in functional and performance test automation in Bay Area computer schools. Vladimir can be reached at vladimir100@hotmail.com*

Statement of the problem. According to the Sky I.T. Group, in today's latest web development environments, there are over 7000 possible configurations of Operating Systems, Browsers, Screen Resolutions and other unique characteristics.

How to produce all these different environments? The traditional way is to build a big test lab equipped with various hardware and software configurations supported. This approach has obvious limitations. The created environment is: 1) hard to replicate; 2) hard to maintain; and 3) hard to share for team testing.

Indeed, to replicate a testing environment, you need to buy another set of identical hardware. This is a costly solution. A maintenance problem arises with the software upgrades. For example, if you upgrade Internet Explorer to a newer version, you are no longer able to restore the previous version in the same environment. In addition, only one SQA engineer can work with a given hardware at a time. This cause prevents parallel testing in the same environment.

Our efficient solution for all these problems is based on use of Virtual Machines with pre-installed commercial functional test automation tools. This approach has the following advantages: 1) Virtual Machine is software; 2) you can easy replicate any environment just by copying the VMware from one computer to another; 3) you have a separate VMware image for each software upgrade; 4) copying of Virtual Machines decouples team testing, all SQA Engineers work in parallel with their personal copies of Virtual Machines.

We used VMware Workstations from *VMware, Inc.* to create test environments with different Operating Systems. The VMware software allows multiple Operating Systems and software applications to run concurrently in Virtual Machines on a single Intel-based computer. The functional test scripts were developed with WinRunner from *Mercury Interactive*. The whole test framework was written in the WinRunner's Test Script Language (TSL).

We created a library of Virtual Operating Systems (VOSes) for configuration testing that contained WinRunner pre-installed on each VOS. That library copied to each test computer. All test scripts were stored for sharing in a central repository on a network and were accessible to any regular Operating System and all VOSes. Another important characteristic of our architecture is that we separated scripts and their data. The scripts are reusable without changes on any test machine. On the contrary, all test machine specific data are stored under another network directory different for each test machine.

For a startup company with limited recourses and budget, test automation is the only choice to assure quality of the product for many different configurations. The Plaxo Contacts application from Plaxo, Inc. is a new tool that enables you to automatically update your address book on different mailing clients and the web. It should work with various versions of MS Outlook, MS Outlook Express, and other mail clients on different Operating Systems and browsers. We support about 30 different configurations. This number will grow in the future. The goal of this paper is to present architecture for an efficient flexible automation framework for configuration testing developed and deployed at Plaxo.

How Does It Work?

General picture. The TSL program that runs within a regular Operating System on each test computer and controls the VOSes on that computer is called *VMware Driver*. The function of the driver is to start a VOS and wait for a response indicating that all the tests on that VOS are completed. Then the driver runs the next VOS from the list of VOSes written in its data file. An example of the VMware Driver data file is:

1=Windows 2000 Professional

2=Windows NT

3=Windows XP Professional

It specifies what VOSes and in which order to run on that test machine.

Once a VOS is launched, it automatically starts WinRunner with a script called *Bootup Driver*. The Bootup Driver reads its data file that contains a list of test scripts to perform for the VOS and executes them. When the tests are finished, the results are generated on a local network share in order to be available for a later review in a regular environment. The results for each VOS are stored in separate directories. Then, the Bootup Driver shuts down the VOS, and passes control to the VMware Driver (Figure 1).

All VOSes run in a consecutive order, with no two VOSes running concurrently. If you need to test another version of the application, you can either update an existing VOS image, or create a new one and add it to the VMware Workstation list (Figure 2).

We separate scripts and machine-specific data. This way, all scripts are reusable. They are shared from one central repository. At the same time, each test machine is associated with a separate repository containing its unique test data.

By convention, we store the scripts on a logical network drive Q and all data on a logical network drive Z. The drive Z contains the physical machine's name in its path. The mapping to Z is different from one test machine to another. For example, test data for a computer called *Vladimir* is stored in the directory: *\\Test_Machine_Data\Vladimir\*, which is the drive Z for *Vladimir,* but for another computer *Tony,* the data are in the directory *\\Test_Machine_Data\Tony\,* which is the *Tony's* drive Z (Figure 3). On the other hand, mapping to Z is the same for all VOSes running on one test machine. The structure of the data directories on Z is identical for all virtual environments.

Because of that mapping, all scripts are always loaded from the drive Q and they are designed to always read data from the drive Z defined at the VOS setup stage.

How to Start WinRunner And Bootup Driver within VOS?

There are two ways to start WinRunner together with a predefined script (Bootup Driver) within a VOS.

The first option is to use a registry entry *[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run],* where you specify a command line under a custom key, like *WinRunner=C:\Program Files\Mercury Interactive\WinRunner\arch\wrun.exe -t Q:\Bootup_Driver -run -verify results_directory*. The drawbacks of this approach are: you can modify the command only within the VOS registry, and you must restart the VOS in order for the new settings to take place.

The second option is much more flexible. For each VOS, the VMware Driver dynamically generates a batch file *current_startup.bat* on the drive Z. It contains a command to launch WinRunner with the Bootup Driver and put the test results in a specified directory. We adhere to convention to name the result directory by the name of the VOS in the VMware Workstation window (Figure 2).

At the VOS image setup stage, we create a shortcut to the *current_startup.bat* within the VOS Startup menu. When the VOS starts, this file will be executed automatically. We can also easily edit the batch file on the drive Z without opening the VOS. To run any new batch commands on the opened VOS, you do not need to restart it. All you need to do is to execute that batch file through the VOS Startup menu.

When all tests on the current VOS are completed, the *current_startup.bat* file is deleted. It will be regenerated for the next coming VOS.

You also need to configure each VOS to start without requiring user intervention such as entering a password.

How to Shut Down VOS from the Script?

We pre-installed a special, Operating System-dependent shutdown utility on each VOS image. To shut down a VOS programmatically from the Bootup Driver script, we simply invoked the corresponding utility.

There are many such freeware utilities available over the web. At Plaxo, we used the *Psshutdown* from SysInternals (www.sysinternals.com) for Windows 2000, XP, and NT, and *Quick ShutDown* from Shatran Software (www.winutility.com) for Windows 98.

How to Synchronize Drivers in Regular Environment and VOSes?

There is an architectural challenge on how to inform the VMware Driver that the Bootup Driver in a VOS finished running all tests, and the VMware Driver can invoke the next VOS. We propose the following solution (Figure 4).

We created a directory called *Sync* on the data drive Z. Before the VMware Driver launches a VOS, it creates an empty file with a pre-defined name there. When the Bootup Driver completes all scheduled tests, it renames that file to *finished*. Meanwhile, the VMware Driver is running in a loop checking every 5 seconds whether the synchronization file was renamed. During the idle time, the regular Operating System passes control to the running VOS to do its job. When the VMware Driver spots that the synchronization file has been renamed, it closes the current VOS, and starts another one from the list.

How Do Bootup Driver and Test Scripts Get Their Data?

We specify a list of tests to perform for each VOS in a separate text file. This file has a name given to the VOS in the VMware Workstation window (Figure 2). For example, we have a *Windows 2000 Professional.txt* file for the VOS image named *Windows 2000 Professional*. These files for all VOSes are stored and edited in a dedicated directory on the data drive *Z*.

Before running a VOS, the VMware Driver reads its name from the list of VOSes for configuration testing. Then it copies the VOS text file of the test scripts to run to a predefined file *current_batch.txt*. Meanwhile, the Bootup Driver is programmed to always look for this file to determine what set of tests to execute. This way, the Bootup Driver is VOS-independent and completely reusable. When the tests on the VOS are completed, the VMware Driver deletes that file. Then it goes to the next VOS in the list and creates a new batch file with the tests to run for that new coming VOS.

Each script has a data file associated with it. By convention, the script's data file has a name of the script. This allows the script to easily find its data file. All data files are stored at a predefined location on the drive *Z*.

How to Organize Test Results?

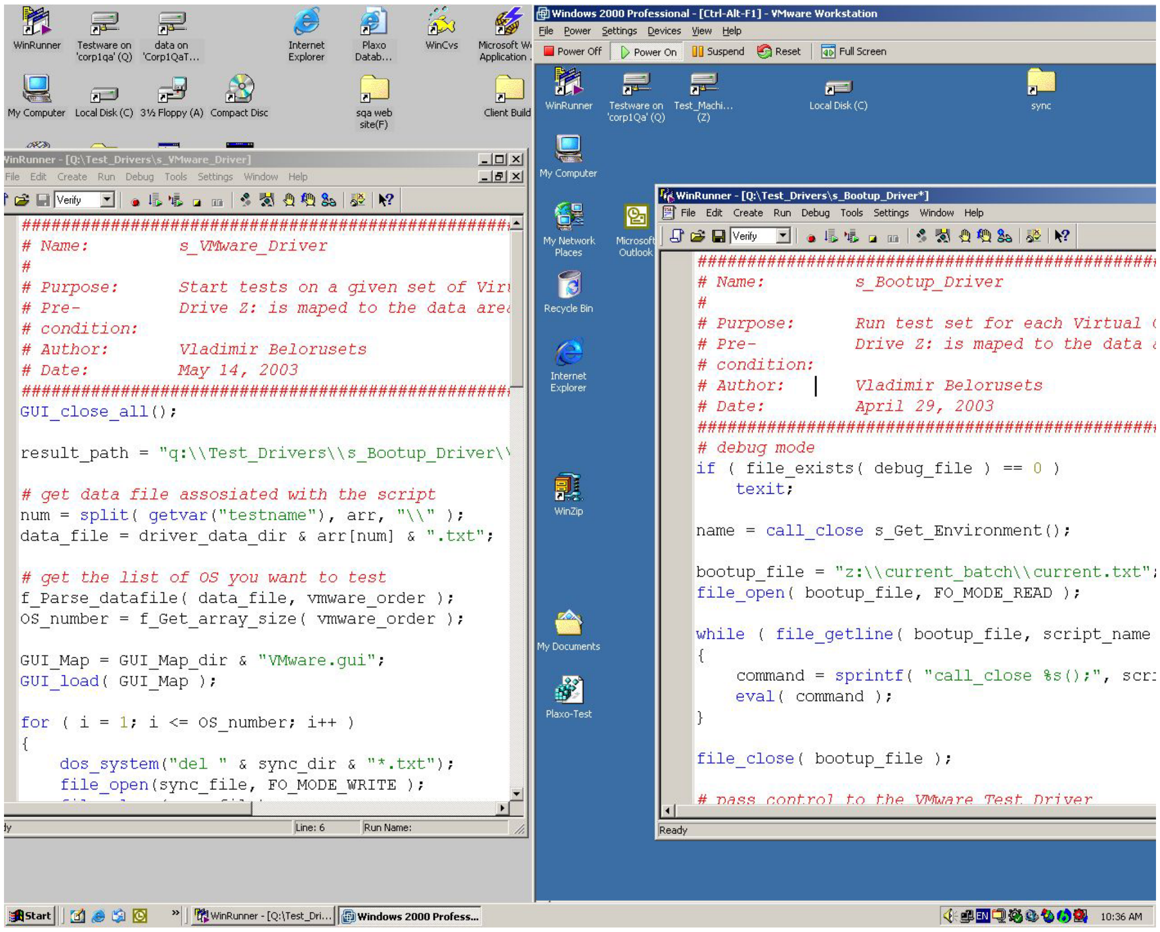
WinRunner can generate test results in two forms: either in ASCII format, or in the WinRunner's proprietary format that displays the results in nice colors and graphics. Since the first option is a plain text, WinRunner allows you to specify the result directory at any location. The big drawback of this approach is that you cannot compare any graphics and it's inconvenient to parse a large result file for specific information.

The second and preferred option requires placing the result directory under the directory of the script being executed. Since the control script running on each VOS is the same reusable Bootup Driver, the test results for each VOS are generated under that driver directory on the drive *Q*. For example, the test results for the VOS named *Windows 2000 Professional* will be generated under the following directory: *Q:\Bootup_Driver\Windows 2000 Professional*. These results can be easily analyzed with the WinRunner's Test Results utility.

There is still one minor problem. Each time we run the same VOS, the previous results will be overwritten with the new results. We solved this by attaching a time stamp to the result directory name.

Conclusion

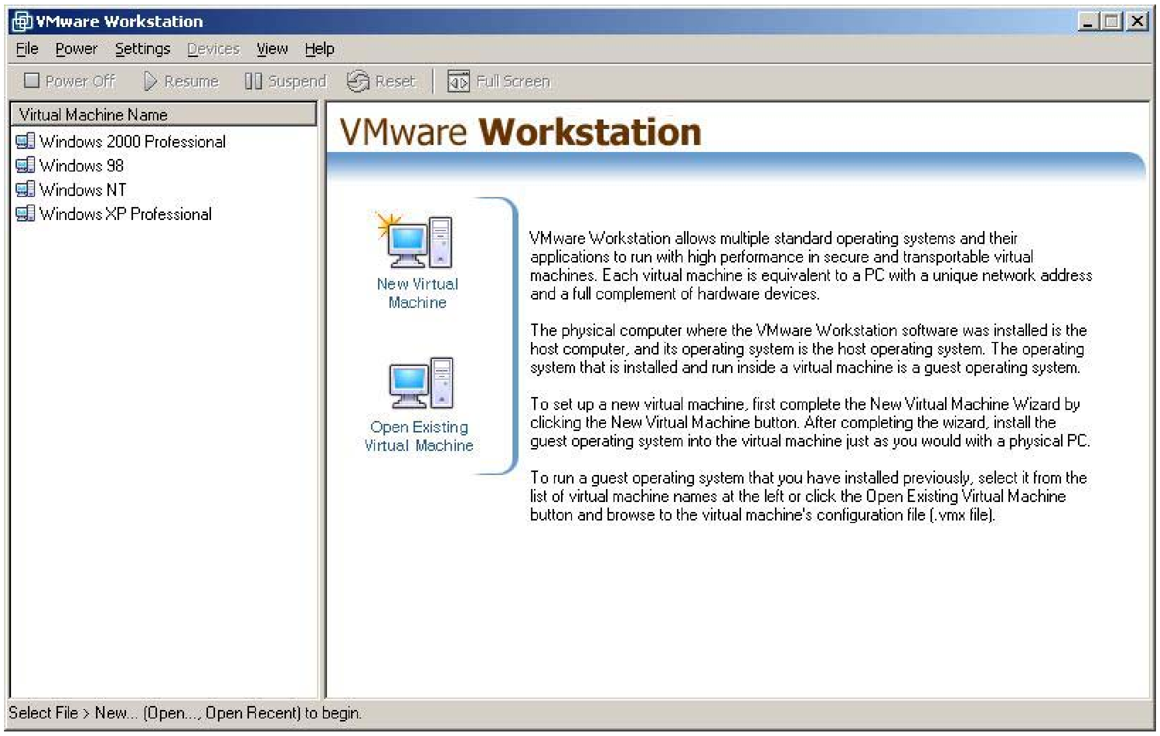
The presented configuration test automation framework was implemented and successfully used at Plaxo, Inc. It was part of a continuous integration process to validate nightly builds. The Plaxo Contacts application was tested with various versions of MS Outlook and MS Outlook Express on Windows 98/2000/NT/XP. The deployed test automation architecture allowed the Plaxo team to test over 30 different configurations on one test machine.


WinRunner - [Q:\Test_Drivers\s_VMware_Driver]
Windows 2000 Professional - [Ctrl-Alt-F1] - VMware Workstation
WinRunner - [Q:\Test_Drivers\s_Bootup_Driver*]
Line: 6
Run Name:
Ready
10:36 AM

```
##################################################
# Name:         s_VMware_Driver
#
# Purpose:      Start tests on a given set of Vir
# Pre-          Drive Z: is maped to the data are
# condition:
# Author:       Vladimir Belorusets
# Date:         May 14, 2003
##################################################
GUI_close_all();

result_path = "q:\\Test_Drivers\\s_Bootup_Driver\\

# get data file assosiated with the script
num = split( getvar("testname"), arr, "\\" );
data_file = driver_data_dir & arr[num] & ".txt";

# get the list of OS you want to test
f_Parse_datafile( data_file, vmware_order );
OS_number = f_Get_array_size( vmware_order );

GUI_Map = GUI_Map_dir & "VMware.gui";
GUI_load( GUI_Map );

for ( i = 1; i <= OS_number; i++ )
{
    dos_system("del " & sync_dir & "*.txt");
    file_open(sync_file, FO_MODE_WRITE );
```

```
##################################################
# Name:         s_Bootup_Driver
#
# Purpose:      Run test set for each Virtual
# Pre-          Drive Z: is maped to the data
# condition:
# Author:       Vladimir Belorusets
# Date:         April 29, 2003
##################################################
# debug mode
if ( file_exists( debug_file ) == 0 )
    texit;

name = call_close s_Get_Environment();

bootup_file = "z:\\current_batch\\current.txt";
file_open( bootup_file, FO_MODE_READ );

while ( file_getline( bootup_file, script_name
{
    command = sprintf( "call_close %s();", scr
    eval( command );
}

file_close( bootup_file );

# pass control to the VMware Test Driver
```

Figure 1. VMware and Bootup Drivers running concurrently.


VMware Workstation
File Power Settings Devices View Help
Power Off Resume Suspend Reset Full Screen
Virtual Machine Name
Windows 2000 Professional
Windows 98
Windows NT
Windows XP Professional
VMware Workstation
New Virtual Machine
Open Existing Virtual Machine

VMware Workstation allows multiple standard operating systems and their applications to run with high performance in secure and transportable virtual machines. Each virtual machine is equivalent to a PC with a unique network address and a full complement of hardware devices.

The physical computer where the VMware Workstation software was installed is the host computer, and its operating system is the host operating system. The operating system that is installed and run inside a virtual machine is a guest operating system.

To set up a new virtual machine, first complete the New Virtual Machine Wizard by clicking the New Virtual Machine button. After completing the wizard, install the guest operating system into the virtual machine just as you would with a physical PC.

To run a guest operating system that you have installed previously, select it from the list of virtual machine names at the left or click the Open Existing Virtual Machine button and browse to the virtual machine's configuration file (.vmx file).

Select File > New... (Open..., Open Recent) to begin.

Figure 2. List of VOSes under test.

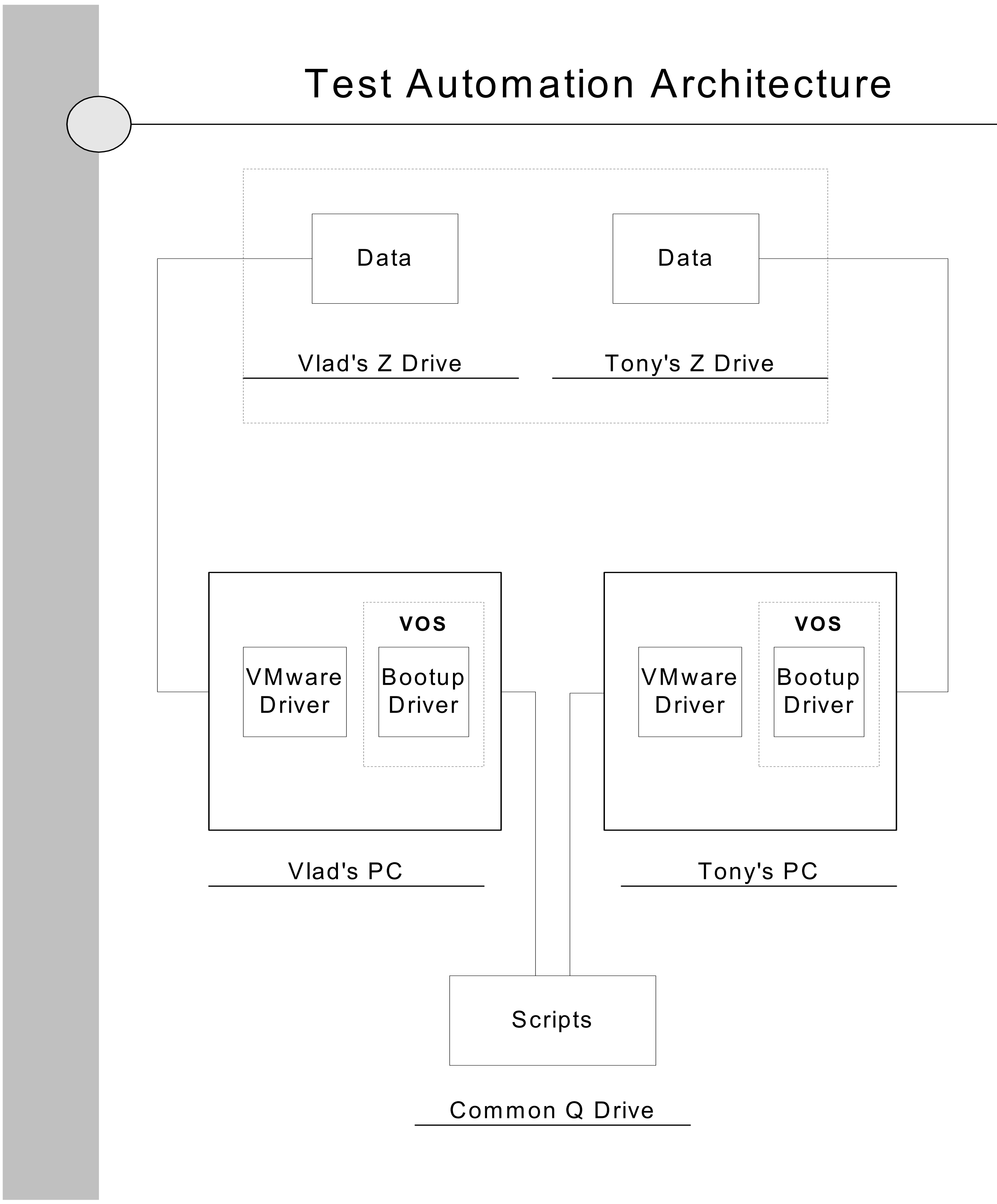


Figure 3. Test Automation Architecture.

Synchronization Diagram

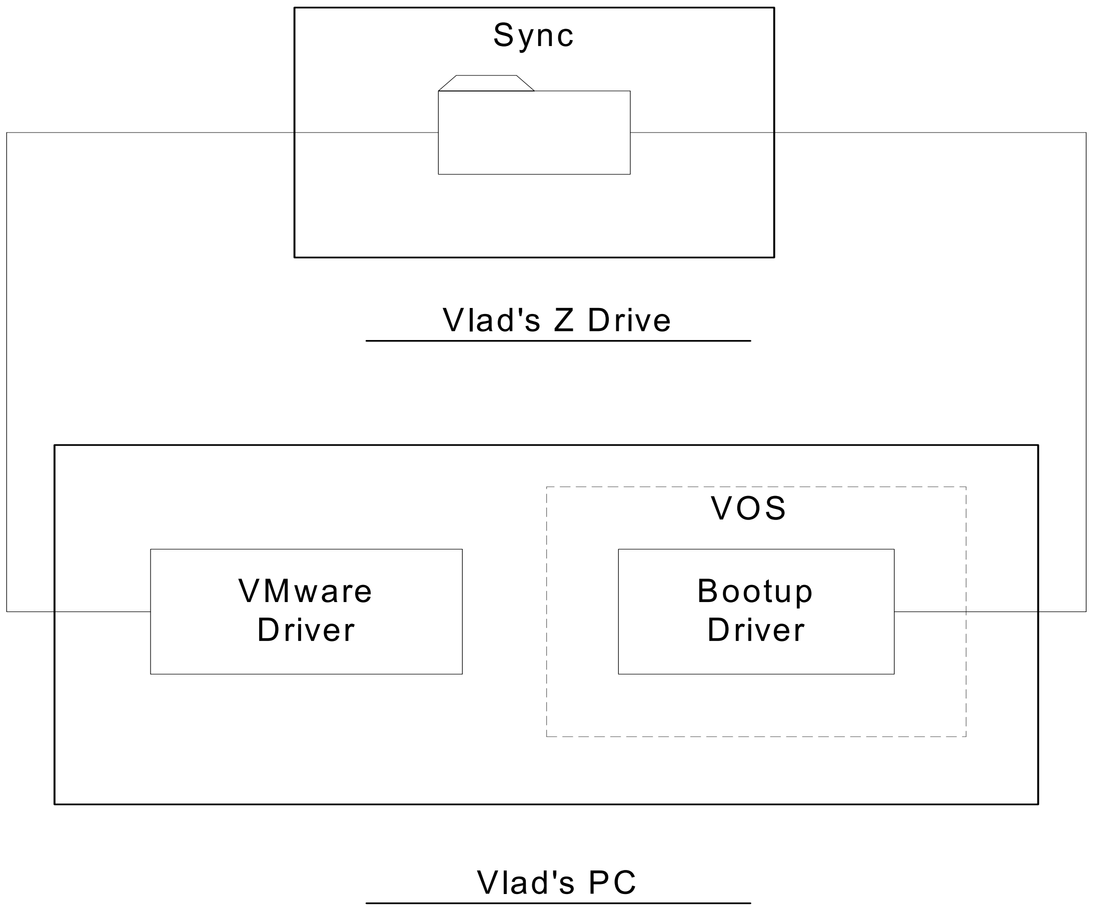


Figure 4. Synchronization Diagram