Nicole Ross

Senior Project:

OEM Validation Library

Final Report

**1.0 Introduction**

1.1 Introduction

I chose to develop a library that will be abstractable enough to attach to any original equipment manufacturer(OEM) test plan. Before I explain what I have done I think it is important to give an explanation of the history and purpose of this project. I will explain the OEM validation (testing) process before the library and then the OEM validation(testing) process now, since the library is currently in use.

1.2 OEM Validation History

Intel would run their own sets of validation and work on finding and fixing bugs. This is accomplished through execution of Test Cases. Test Cases are a series of manual and some automated tests. After they would finish a set number of development cycles for wireless cards they would release them to the market. Computer manufacturers would put Intel's wireless cards in their systems and do their own validation for compatibility. All of these test are manual. Manual test are very slow and not efficient. The good thing about the computer manufactures doing their own tests is that their engineers think differently. They do a series of different test and find different bugs that where undetected on Intel's side.

My current boss, Munir Chhibber was the guy on Intel's side that dealt with the other companies. He would get their reported information and try to tailor Intel's test to include what the OEM was testing on as well. This was all done by one person and was not being done efficiently because of this persons many roles they played at Intel. They needed more man power. I heard a lot of talk before returning back from my internship. The talk was about creating another small group that dealt with OEM test only. I had ideas for this and discussed it as a possible project when returning to NMU if I could actually get the lab up and running.

1.3 OEM Present Validation

There was a group created at Intel just to deal with OEM's validation. They have multiple different type of OEM machines and test the Intel Wireless Card before shipping it off to the market. This group uses the same exact test that the other companies test with but instead of manual test they are making an effort in automating these test. They get the test in various forms I have listed an example below:



TABLE I: AN EXAMPLE OEM TEST CASE

The table above has three columns. The first column is the step number. The second column is the actions that needs to be performed (automated). The third column is the expected results. The expected results in this example seems elementary but with other technologies such Bluetooth a tester might not now know what the expected behavior is for example: Have a computer search for Bluetooth devices, while searching turn off the Bluetooth switch. Is it suppose to throw an error, close gracefully, or continue running. A tester would have no clue in knowing the expected behavior without having their hands in the stack development.

When I started to code I was introduced to many terminologies that I have never heard of, such as "SUT" this is System Under Test. Intel uses the terminology "DUT" (Device Under Test), they are interchangeable terms. I did not know what Windows Zero Configuration(WZC) was right away. I discovered it was used to configure wireless cards. WZC is used when the Manufacturers Wireless Utility is not installed.

I had no clue what "S3" was. I did some research on this and found they were power states. I guess saying S3 is shorter than saying standby or sleep. I researched what each S-State meant and what was going on in the hardware and software that separated it from other S-States. Doing many test that have various power states are referred to as ACPI Test (Advanced Configuration & Power Interface). ACPI is an open industry specification that was co-developed by Intel and several other companies. ACPI establishes standards for power management and OS-directed configuration.

The new team at Intel take these test and develop automation for them and then performs the test on the appropriate hardware. Many of these test follow a similar pattern as far as what type of code it would need to automate it. I decided if I could create a set of functions that most of the OEM companies would use it would save a lot of time for validation engineers, interns, and NMU Lab Engineers. The OEM Library was born from this idea. To be an abstractable library of functions to help validation engineers with testing OEM products. The library can be attached and from there all the code I have created is just a function call away.

**2.0 Project By Phases**

2.1 Phase 1

**2.1.1 Analysis**

Phase 1 was considered my beginning phase. I did a lot of analysis and research during this time period. I needed to get more information on what the library would be used for, such as what type of test would be done using this library. I had to gather up what topics I would have to learn, and what tools I would need to complete this project. I also felt it was important to do the test manually before I built a library that could be used to automate them. Through doing these test manually I learned a lot of different networking terminology I list in the section "Topics Learned".

 I first had to learn what ITE was. ITE stands for Integrated Test Environment. I discovered it was a tool to help with the validation process. ITE is a program that Intel uses to run test sets on a specific device. I had to learn how ITE worked without any training. This was a hard task. It is not a very easy program to use and I had a lot of problems with its different quarks.

I had to set different system up with ITE. I put it on my development machine, my Devices Under Test (DUTs), and on the server I have running in the lab. I learned VB Script. It is a very easy language. I can see why they chose this language for the automation. It took me about 30 minutes of reading and then I could code a good chunk of what I need to code. There were things along the way I had to look up how to do it specifically in VB script such as how to Wait(), search for first occurrence in an array, capitalize all letters of a parameter, and how to actually return a value from a function.

Much of my work in the beginning phase was trial and error. There was no manual for what I was doing so I had to just poke and prod at the program to see how I could get it to do what I wanted it to do. There was a lot of research done during this phase. It was very important that I did the test manually first before attempting to automate them. Doing this really opened my eyes to what I did not know. I went further into detain about this in the "Learned" section.

**2.1.2 Setup**

Part of this project required some initial setup. I needed to learn the program Ghost to prepare each "Device Under Test" to test the OEM library. These DUTs are Toshiba Laptops that were provided by Intel. They run a small program on them called "ITE Executor" that allows the ITE Editor (the development environment) to connect to that laptop to do whatever I program it to do. I also needed to set up the physical network. These DUTS needed to be on a separate network then NMU's . John Marra (NMU's Network Administrator & Batman Fan) created a VLAN for the lab to have our own block and network on campus. I was very unfamiliar with this but quickly learned its purpose as I started to do more with the networking aspect. I set up switches, Access Points, Power Distribution Units, computers, and a server to run on this network. I also needed to create an SVN repository to allow all Intel and future lab workers to have access to the library. SVN stand for subversion, it is a place on a server to hold, organize, and give permissions to certain code. This helped Intel track my progress and gain access to the latest library easily.

Getting ITE to work on the machines and talk to other devices was not easy. I had many errors that occurred quite frequently. I had to discover that navigating ITE was an art in itself. If you are to attach a DUT you need to attach it in your local widget, add the appropriate wrappers to it, and then assign it as a resource. There are multiple fields that need to be filled out for each reference that you refer to such as Access Points and DUTs. You have to connect ITE through the Ethernet IP Address. One problem that I had was many of the functions required me to change the IP Address of the WiFi adapter. I was changing the IP Address of the Ehternet Port instead of the WiFi and this was causing me to lose connection with ITE. Many bugs like this delayed production and helped me appreciate hand-on training.

2.2 Phase 2

**2.2.1 Design**

Once I had a good understanding as to what I was doing I needed to create a changeable design as to how I was going to build the library. It needed to be well organized, abstractable, and easily portable. I wanted each function call to be located in an appropriate file with an appropriate file hierarchy (Directory Names). I needed to design the functions to work in many environments, different DUTs, different OEM tests, with no information hardcoded into the functions, such as IP address and device names. The library needed to be built so that it can be attached as a single "widget" to be used with OEM test plans. A "widget" is a GUI attachment in ITE that has a set of configurations made for that widget to be used in multiple test sets. You can make a widget that other people can use with devices attached to it and various wrapper libraries attached to those devices. This concept I did not understand at first. I never heard of wrappers and marshalling. This was explained to me by Batman also know as Andrew Hawker. Andrew is one of the developers of ITE. He was a graduate from NMU and he has been assisting me in understanding how ITE works frontend and the backend.

I chose to follow a certain pattern for developing. Instead of just sitting at a computer and hacking away at the code I drew pictures of the libraries hierarchy. I built what I refer to as the frame or skeleton of the library. I created the files and linked them in the library. I built directories and stored the appropriate files in the well thought out naming convention that I came up with. I then built empty functions that I wanted to see implemented in the library. Taking my time in this phase helped me be a more directed coder when I sat down to code. I have included a hierarchy diagram that helps show the relationships of the different files in the library below:



FIGURE I: HIERARCHICAL DESIGN OF OEM LIBRARY

2.3 Phase 3

**2.3.1 Code**

The coding looks easy but that is not the case. The amount of research and troubleshooting that went in to certain lines was a hidden surprise to me. I would code a function and then test it with good input and then bad input. This is how I verified my functions as I coded. I would have to do research with every function that I created because it called or referenced things that I was never introduced to. I have included a snippet of code below:

Function CreateAdhocNetworkWPAAES (DUT, SSID, WPAkey)

 DUT.Profile.DeleteAllProfiles()

 Dim dutCreateProfile

 Set dutCreateProfile = DUT.Profile.CreateProfile.WPA.Personal.AES(SSID,SSID, WPAkey, 1 ,2)

 If Not dutCreateProfile.Passed Then

 ITE.TestLog.Results.Error "Failed to create an Ad-Hoc Network on DUT: " & dutCreateProfile.Description

 Exit Function

 End If

 Dim dutConnectProfile

 Set dutConnectProfile = DUT.Profile.ConnectProfile(SSID)

 If Not dutConnectProfile.Passed Then

 ITE.TestLog.Results.Error "Failed to connect DUT to Ad-Hoc Network: " & dutConnectProfile.Description

 Exit Function

 End If

 ITE.TestLog.Results.Pass "Successfully Created Adhoc Network with WPA/AES."

End Function.

FIGURE II: EXAMPLE FUNCTION LOCATED IN WIFI\_PROFILE

I will go through the above example and explain what I had to learn, do, and typical problems that occurred. I had troubles at first because I was creating many functions that created many profiles on the DUT's, this causes problems. Many of the functions started out with less parameters, but the more OEM test cases I read the more I realized the importance of making them abstract. This means more parameters. I added parameters to all the function of the values that changed from test to test like the DUT object, the SSID, and the key.

I experiencing many errors in creating the same Ad-Hoc profiles over and over again. This causes problems in windows. I figured out I had to somehow delete all pre-existing profiles before I created a new one. That is where the " DUT.Profile.DeleteAllProfiles()" came from. When " Set dutCreateProfile = DUT.Profile.CreateProfile.WPA.Personal.AES(SSID,SSID, WPAkey, 1 ,2)" call is made an ITEReturnValue is returned and you can pull description data from that ITE object. I had to play around with this in other functions because sometimes it is returned wrapped in something and I had to figure out how to strip it from its wrapper to get a pass or a fail. Once the profile was created it would (hopefully) output a green bar in the ITE status window with a description. If it failed it would be red and have a failure description.

After the profile creation the DUT would be able to connect to that specific profile. The function call listed above would have to be called on two DUT's and then they would create a Ad-Hoc network in between them.

2.4 Phase 4

**2.4.1 Test**

I tested functions as I created. I also created a phase in my project to make sure multiple test where performed on all functions before I passed this library off to Intel. I wanted to make sure the functions passed with proper parameters. I would then try to make them fail with bad parameters. I had one function that passed with bad parameters. This function was an Iperf traffic function. I did not understand why it passed with when I was passing in "0" as an IP for the server. How was traffic being ran if the two computers could not connect to each other. I found out through research that some functions calls will just pass whether they worked or not. What this meant to me was I had to find another way of testing.

I started to use a tool called Wireshark. This is a tool used to sniff network traffic. This tool is commonly used to "sniff" other people's internet packages. I had to figure out how to work this tool and use it to sniff the packets that was or was not sending over Wi-Fi. I found I was sending the packets over the Ethernet port. I had to modify the function call to send traffic over Wi-Fi. I also created multiply traffic function that tested the traffic. This is helpful in troubleshooting and can later be used as an analysis tool.

2.5 Phase 5

**2.5.1 Release - Evaluation**

The release - evaluation process came when I was interdicted to the team lead that does the OEM testing. He reviewed my library and was very impressed. His team started using it right way. Munir was happy with the abstactability and the amount of work that was done. This project can be continued on by future Lab Engineers. I have started to train others in the ways of ITE and have given them an introduction to the OEM Library. They will be adding to it as more test come in. There is talk of doing more Bluetooth and Wi-Di testing. If this is the case the library will need to be expanded to include further functions that can support these technologies further.

**3.0 Reasons for Choosing Project**

This project was exciting because of all the interesting things that I could learn. I chose this project because it was something that I have never done before. I thought this would help my career more than doing a project that I already knew how to do. My career is starting in Enterprise Application Engineering and I wanted to work with enterprise products to do similar tasks to prepare me for life at Intel. I learned what goes into a large scale project and having customers that have deadlines and expectations. I learned how to meet those expectations or explain properly why I fell short for that certain time period.

I am also very interested in making things better. I believe a product should meet customers' expectations and exceed other manufacturers. The competitive nature in me makes me want to find bugs and prefect the product that I am working. If everyone shared this attitude the product would become superior to another manufacturer. I was very passionate about learning about this process while I was at Intel. I was so intrigued by this I would to meetings that where in different groups to get a perspective of the products lifecycle from cradle to release. During one of those meeting I attended I met a met a man named Munir. We spoke about the OEM process and how it plays into Intel Mobile Wireless Group. We started meeting once a week and started to discuss plans for NMU and Intel together. This is how the NMU Intel Lab came into existence and the seed was planted for this project.

**4.0 Topics Learned**

Through most of the troubles I ran into came the most learning. Along with not having a clue to about 90% of the terminology I did not know how to do the manual test that I was trying to automate. I listed only some of the technologies, terminology, and skills I gained during this project below:

* Cisco OS calls
* Networking protocols and terminology
	+ Add-Hoc
* Programmable switch
* Iperf
* Collaboration with client over want, needs, and timeline
* Code under someone else's specifications
* Comment under someone else's specifications
	+ To implement Robo Doc
* VB
	+ regExpParseString - regular expressions are not that easy in VB
	+ Wait () to allow other functions to catch up before preceding
* Switched Rack Power Distribution Unit
* Wireshark packet sniffing
* Code version control using SVN repository
	+ Creating a repository on server
	+ Learn skills and techniques for comparing code with previous versions and documenting
* Corporate Coding guidelines
* Effective use of creating a timeline for a project that will continue after my departure
* Utilizing abstraction layers
* Validation Engineering
	+ Building test cases from scratch
* AP programming
	+ Learning Cisco's AP system calls
* APC Switched Rack Power Distribution Unit (PDU)
	+ How to set up from scratch
	+ How to set up from a previous installation (much harder)
	+ Local configuration as opposed to networked configuration
	+ Read a lot of the user guide for command line interface
	+ Needed to learn how to send command line commands to the PDU using telnet through Another program

Some of the problems that I had during the project development was some of the hardware that was given to me was not in the appropriate modes. AP's need to be in autonomous mode meaning they did not need a controller and I had to figure out how to get it into that state. The power distribution unit was assigned to another IP address. I could not figure out how to dial into it to change the IP address. I ended up patching two cables together. On one of the cables end was an RJ11 (phone) to standard serial(male) the second cable was standard serial (female) to USB. I had to search the internet for a "hacked" driver to get the USB/Serial conversion to work. I then assigned a port in the device manager to the serial device, enabled telnet using windows, and dialed into the device.

**5.0 Conclusion**

I am very happy with my project. I know I learned a great deal along the way; along with contributing to future plans with Intel and NMU. Things that I would have changed is more training on certain topics. I would have liked more hands on training with ITE so I could focus more time on wrappers and abstraction layers. I learned the importance of communication towards all clients. I hope this report conveyed my project better and I look forward to demonstrating the libraries functionality.