TJHSST Computer Systems Lab Senior Research Project Automated System Testing 2008-2009

Ian Garrett

June 8, 2009

Abstract

Around the world, many companies require testing for various projects. This process can take numourous hours of manual labor, which in turn costs companies a lot of money. With automated software, companies do not need to spend human resources to manuelly test. In the business world, there is not much room for time or money to be wasted. This project reduces the time spent in testing with the use of software that can be obtained easily. It sets up a system in which one client system can test multiple applications on many server systems with a high consistancy. The one client system implements various automation tools to accomplish the task. The practical use of this project is to reduce the time spent on testing while using automation tools that anyone can obtain. The project will show that the manual testing that originally takes hours to test on the one or two other systems can be reduced to just minutes using one system.

Keywords: automation, client, server, testing

1 Introduction

1.1 Scope of Study

This project implemented a few significant applications to test it. TightVNC and VNCRobot play critical roles within the project. I have set up TightVNC,

which creates either a server or a client onto the system. This makes the server's system available to the client machine. When the client opens up a viewer, the client will be able to access all of the servers that it is connected to. VNCRobot plays a role in helping the automation by being able to capture actions made on the computer.

Although this project can be done with many systems, this project will be limited to two systems. This is to not overwhelm the project as setting up systems takes a long time, and the main objective of this to prove automation is effective as a tool by reducing test time, not expansion into many systems.

TightVNC, creates a VNC envirnonment. This is an application that is not specfically designed for automation, but just for any purpose of use through VNC. Having a VNC environment means that two (or more) computers are connected and be accessed remotely. The importance of this application is to be able to access another system from a single system. TightVNC (as well as any VNC environment) invovles two components. The first is the client, and the second is the server. Server systems are controlled by the client system, which is responsible for any work being done remotely. While alone TightVNC is not automation, it plays a vital role in the automation process in this experiment.

The other application that is key for success in this project is VNCRobot. Like TightVNC, VNCRobot implements a VNC environment. The difference is in that VNCRobot can be used to control another system. By itself, VNCRobot is not test automation as it can be used (as is more commonly used) for other purposes. VNCRobot is significant in my project because it has a capture feature in which the user can record movements. Using VNCRobot as well as coding in the movements for testing helps create the automation.

An advantage to the method I used to create the test automation is the ease in which it can be created. The applications that help with the automation are easy to obtain. This creates a cost effective means to automate tests for future software. Currently, automation is a highly expensive process so to be able to create automation with mostly free means is important.

1.2 Expected results

This project is expected to have a reduced testing time using automation. The system that is set up should include a client with a server and implement easily accessible applications during the process. There should be a drastic

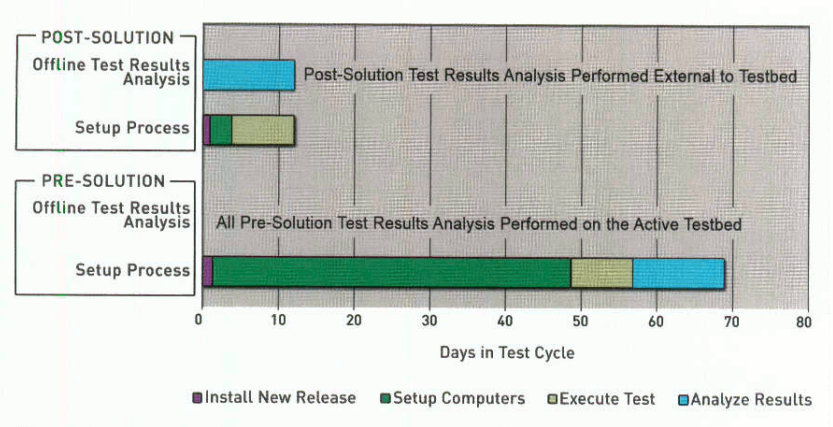
difference between the manual testing time and the automated testing time.

This project may be later extended to a large-scale system. If the client were to control a hundred servers, instead of another one, it may change the outcome of the project. Therefore, the testing of systems through automation can prove to have even more purposes.

2 Background and review of current literature and research

Companies have been trying to impletment automated testing within their companies for years. Many common problems that halt the continuation of this is a lack of knowledge in the area and the long time that it takes to set up automation. Years ago, automated testing was not practical for use due to the long times it took to set up an automation system. Currently, there are many tools for automation that makes automated system testing not only simple but extremely quick to set up. The increased knowledge and simplicity of the automation process in key in automated system testing.

Figure 1: Gonzalez "Automating Software Configuartions in Test Labs"



A similar experiment to this project was done by Gonzalez and Spade. They also chose to attempt in reducing testing time to reduce cost of testing. However, they did not use easily accessible means of testing. The data (Figure 1) shows that the means in which they tested did in fact reduce the time from almost 70 hours of total set up and testing to slightly over 10 hours to do so. My project will not include set up time as this one did.

Often people do not realize the significance of automation and overlook it. Unfortunetely, when people get around to testing there are many obstacles that may occur. The paper titled "Heuristics-based infeasible path detection for dynamic test data generation" realizes the importance of automated tests and its role in reducing cost and increasing reliability of software testing. It also states that there is a large challenge with path-oriented test data generation. The goal of the experiment was to find a way to solve the issue of infeasible program path detection for dynamic test data generation. A key player in solving the issue is the fact that many of the infeasible program paths have similar properties. They use this fact to try and solve the problem by knowing what signs would most likely lead to an infeasible path. This

way, the path can be detected with fairly high accuracy.

Another paper titled "Building test cases and oracles to automate the testing of web database applications" explains how many organizations use web applications with databases to store data but manuel testing of these applications can take an impractical amount of time. The solution to this is to automate the testing but this can even prove to fail due to poor test cases, even if the automation code is perfect. Therefore, the Automatic database Tester was created to generate functional test cases that would be used for database updates. The project that I am doing uses a test case to set direction in the project.

3 Development

For the purpose of the project, I chose to select a random software to test. SeaClear (Figure 2) was chosen as an application because of its potential to show the full effect of automation. SeaClear has many drop down menus as well as input values that can be automated. For example, known valid values were plugged in. Later, known invalid values were plugged in to see if an error menu will appear. SeaClear was not used as a tool of automation in any way but was successful in showing the wide range of what automation could perform.

Image of the SeaClear Program

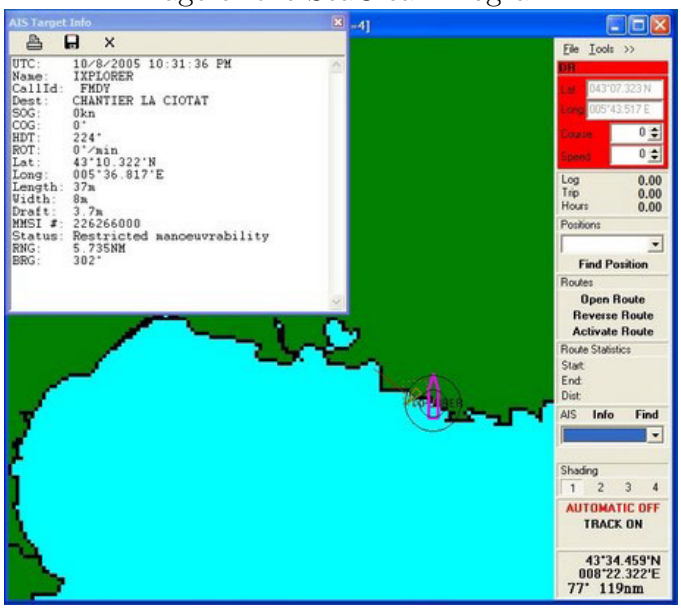


Figure 2

To have data to compare the automation with, I had to manually go through the test case. The test case was 58 lines long and was used to show the different commands that automation can perform. I chose to run through the test case three times. The average of the three tests were about 15 minutes (Figure 2).

The client system will use TightVNC to connect to the server systems. This will create a relationship between one client and one server. I have coded commands into VNCRobot so that not only is the server system accessed, but it is now automated. The SeaClear application (Figure 2) will be tested and the results will be sent back to the client. The test commands have been added in so that the process is now an automated test of SeaClear. The operator, will then be able to view the results within minutes of starting the testing. The data collection will be through the comparison between manual and automated testing. The initial test will be a test case built for a manual test whose time will be recorded. This includes work in SeaClear.

The project's accuracy will be tested by first manually testing the application. After the results have been recorded, the client will attempt to recreate the results using the same testing process. If the results match, the project was successful. This process will be repeated with different types

of inputs, such as valid verses invalid inputs, to show that any input will produce correct results.

The next step in the process is to automate the test case and then run the test case with the automated environment that has been set up. The time for completion of this step will also be recorded. After the initial code was written for the automation, there was a long editing process. The code as is was only good enough to automate the test in the time a person could do it. I had to look at the wait times for each of the commands and set it to an appropiate time.

A Sample of the Code

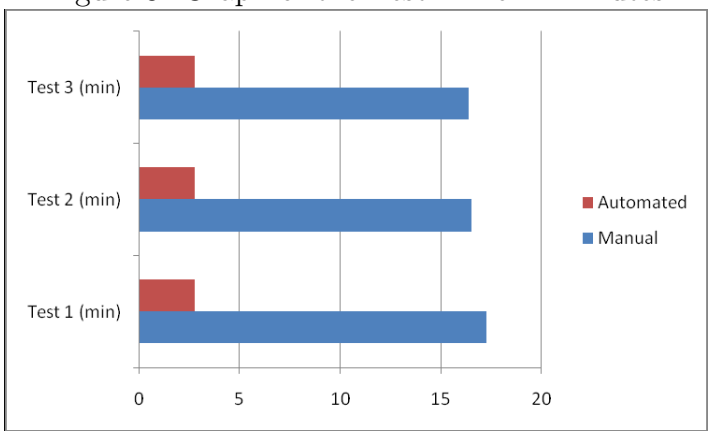
Mouse click to=x:826,y:359 wait=100
Mouse drag from=x:764,y:324 to=x:764,y:323 wait=100
Mouse move to=x:764,y:324 wait=100
Mouse move to=x:764,y:325 wait=100
Mouse move to=x:764,y:324 wait=100
Mouse click to=x:764,y:323 wait=100
Press Backspace wait=1000
Mouse move to=x:832,y:286 wait=100

A large issue with this was the speed at which it went. If I set the wait time all to incredibily fast speeds, the program would act faster than the recieving computer could react. For example, if a drop down menu command was sent at a faster speed than the computer could actually drop the menu down, the program would be sent in the wrong direction later on. In addition, there were many commands that were extra from the recording processs. Therefore, I had to go through all the code and weed out or replace the commands that were not needed.

4 Results

The automation of the testing did not compromise the accuracy of the testing in any way while successfully reducing test time. The data from the main test shows that the testing time was greatly reduced (Figure 3) from an average of 16.5 minutes per manual test to an average of 2.75 minutes for each automated test.

Figure 3: Graph of the Test Time in Minutes



The project also successfully created the VNC environment that was vital for completion. There was one client and one server system.

After implementing the final test, there were a few different results. The test was run three times. The bars representing the manual testing is a result of going through the manual test three times. This consists of going through the entire test case. The bar representing the automated test is a result of goign through the automated test three times. Since there was code involved with automation, only the program had to be run three times to go through the test case. The manual tests varired with each run but the automation was a consistant 2.75 minutes each test.

5 Analysis

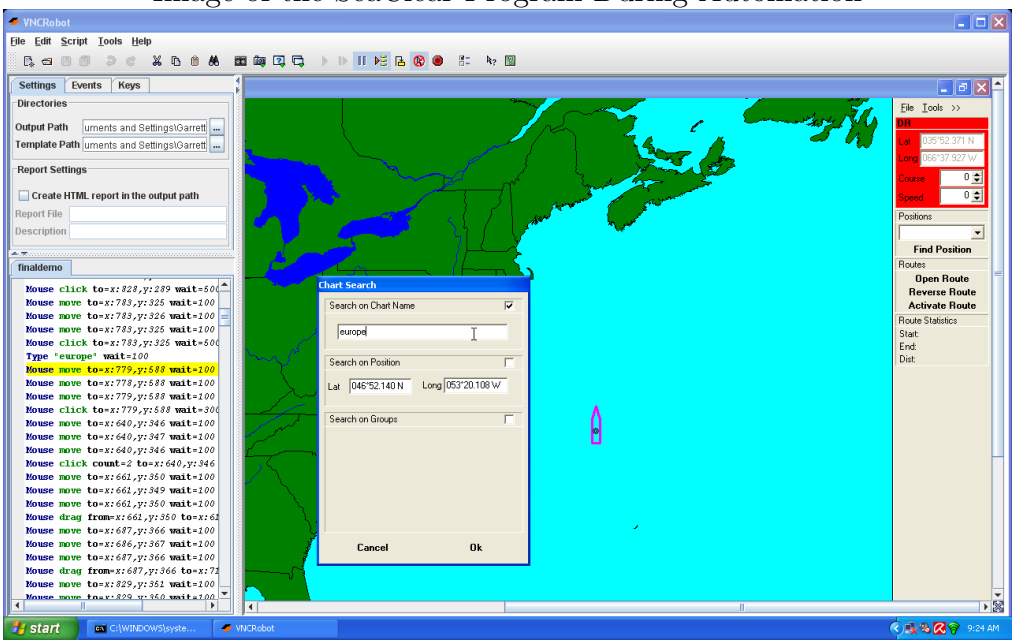
The data collected was in the form of different tests. The main objective was first to have everything running properly. After setting up the environment that was needed to have everything working, I started to test different application before settling for the one that would mainly be used for the project.

One of the simple tests was to see how the automation would work with a word processor. To test this, I had the program open up Microsoft Word and type out the log that was due that day. Through this test I learned that the program could work for a long period of time. This was shown through the fact that I had a full-page log and the program successfully complete it.

Another simple test that was used was to take a screenshot of the stocks and email them to myself then check it. I went to BigCharts.com to check a random stock then took a screenshot. Then, I entered my email and sent the file of the picture to myself. I did this multiple times. I learned from this that the program could be used to repeat tasks, which was good because that was one of the project's main goals.

After it was determined that the automation was working properly, I moved on to the final test. The objective was to see if automated runtimes were significantly lower than manual runtimes. In addition to comparison with runtimes, there was an issue with consistancy. A manual tester will not always be consistant. Many factors can affect the efficiency of a worker while the automated tester will run the same time everytime.

Image of the SeaClear Program During Automation



6 Discussion and Conclusion

The goal of the project, which was to reduce test time in software testing, was successful. The project could go through a test case built by the user without the need of a person to continually run the software being tested. A VNC environment was sustained and the two systems could interact with

eachother. In conclusion, automating software for testing is worth the initial time to set up. The automation increases the efficiency of the test as well as consitancy.

In future experiments, I would like to use more systems. The current version of this project only consists of two systems. This should be able to be expanded to one client and many servers. In addition to extra servers, a later version of this project may include a VNC environment that consists of multiple operating systems. The current version only includes Windows but the VNC environment should be set up on a Linux OS or a Mac OS.

Another future extention to this project is to use only data to test. Currently, I use graphics and gui to represent the testing. There are many programs that do not rely on any graphics to run and therefore can also be tested. If this project is to be extended in this way, the test time would be reduced in even greater amounts than it is currently.

7 Referances

Dhavachelvan, P., Uma, G., Venkatachalapathy, V. (2003, November 14). A new approach in development of distributed framework for automated software testing using agents.

Gonzalez, R., Spade, R. (n.d.). Automating Software Configuartions in Test Labs.

Hawkins, J., Nguyen, H., Howard, R. B. (n.d.). Automated Real Time Testing (ARTT) for Embedded Control Systems (ECS).

Ngo, M. N., Tan, H. B. K. (2007, January 15). Heuristics-based infeasible path detection for dynamic test data generation.

Ramamoorthy, C., Ho, S. (1975). Testing Large Sotfware with Automated Software Evaluation Systems. Retrieved September 30, 2008

Ran, L., Dyreson, C., Andrews, A., Bryce, R., Mallery, C. (2006, May 24). Building test cases and oracles to automate the testing of web database applications. Retrieved October 28, 2008