Development of monkey tool for sequential, concurrent and random scenario

Submitted By

Palak Dixit

19MCEC03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2021

Development of monkey tool for sequential, concurrent and random scenario

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Palak Dixit

(19MCEC03)

Under the guidance of

External Guide

Mr. Suraj Haldar

System Validation engineer,

Intel Technology Pvt. Ltd.,

Bengaluru.

Internal Guide

Dr. Swati Jain

Associate Professor,CSE Department

Institute of Technology,

Nirma University, Ahmedabad.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2021

Certificate

This is to certify that the major project entitled **"D"evelopment of monkey tool for sequential,concurrent and random scenario** submitted by **Palak Dixit (19MCEC03)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project Part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Jain

Dr. Swati Jain
Internal Guide & Associate Professor
CSE Department
Institute of Technology
Nirma University, Ahmedabad

[illegible]

Dr. Priyanka Sharma
Professor & PG Coordinator (M.Tech - CSE)
CSE Department
Institute of Technology
Nirma University, Ahmedabad

[illegible]

Dr. Madhuri Bhavsar
Professor & Head
CSE Department
Institute of Technology
Nirma University, Ahmedabad

[illegible]

Dr. Rajesh N Patel
Director
Institute of Technology
Nirma University, Ahmedabad

intel

Certificate

This is to certify that the major project entitled **Development of monkey tool for sequential,concurrent and random scenario** submitted by **Palak Dixit (19MCEC03)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by her under our supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Biswadeep Sengupta

Mr. Biswadeep Sengupta
Manager
Intel technology Pvt. Ltd.

suraj
digital signature
5/17/2021

Mr. Suraj Haldar
External guide
Intel technology Pvt. Ltd.

Statement of Originality

I, **Palak Dixit**, **19MCEC03**, provide endeavor that the key Project named as **"Development of monkey tool for sequential,concurrent and random scenario"** presented by me, towards the partial fulfillment of the necessities for the degree of M.Tech in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school to the best of my knowledge. It is the original work meted out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is revealed or written earlier, except where reference has been created. I perceive that in the event of any similarity found after with any revealed work or any thesis work elsewhere; it will result in severe disciplinary action.

palak

Signature of Student

Date: 18/5/2021

Place: Surat

Jain

Endorsed by

Dr Swati Jain

(Signature of Guide)

Acknowledgements

It gives me pleasure in submitting this project report towards successful completion of my major project. First and foremost,i would like to thank **Dr Swati Jain** who is Associate Professor of Computer Science and Engineering Department branch in Institute of Technology, Nirma University, Ahmedabad for her valuable guidance during this project.

I would like to express my sincere gratitude to **Dr Madhuri Bhavsar**, Hon'ble Head of Computer Science And Engineering Department, Institute of Technology, Nirma University, Ahmedabad for providing basic infrastructure and healthy research environment.

It gives me an immence pleasure to thank my manager **Mr.Biswadeep sengupta**, Manager, chrome validation team, Intel Technology Pvt. Ltd. for his constant support throughout my internship project.

A special thanks to **Dr Rajesh Patel**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his motivation.

I would also like thank all my friends for their help and support as well as faculty members of Computer Science and Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions throughout the development of this project.

- Palak Dixit
19MCEC03

Abstract

Executing the concurrent tests puts a greater emphasis on user's real-world scenario, and also help to showcase the robustness and performance capabilities of Intel platform. Also, with Android container introduced in Chrome OS, it provides opportunity to be included in concurrent test suite to replicate the real-world scenario of running multiple applications simultaneously.

As a part of Chrome system validation team in CCG, i have introduced system level "concurrent test suite". i have observed that when multiple testcases are running concurrently, the system resources like CPU and memory are gradually exhaust, and there are very high chances of system crash or failures.

Technical Key terms: Automation, Monkey, ADB, Concurrent, Random, Sequential

List of Figures

Contents

Chapter 1

Introduction

The objective of this project is to develop a monkey tool that will create user scenarios and check for kernel panic, errors, or issues. It will address the issue of replicate the real-world scenario of end-user as seen in Field Testing with automation. It will be implemented in the chrome validation team with Automation is done using Python Autotest. The expected functionalities of this tool contain sending input events (keyboard, touch), opening URLs, and perform tasks like YouTube playing, graphic website usages, browsing content, install-uninstalling of apps, etc., measuring CPU load, memory load, temperature, controlling time between events, etc.

1.1 Problem Statement

When a user performs heavy lifting tasks concurrently, there will be an exhaustion of the system resources (memory and CPU bandwidth), and which result in failures in completion of tasks.Existing test suites such as MTBF test suite and Android's Monkey emphasize on running only in serial manner, and do not replicate the real-world user scenario.Intention of this work is to fill up this gap.

1.2 Solution

To leapfrog from the traditional sequential stress testing, i have developed a "sequential,random and concurrent test suite". It includes a set of workloads to map user scenario, and then continuously measure the metrics during the execution of the tests and it will be platform and OS agnostic solution (Windows/Android/Chrome). I have observed that when multiple workloads execute concurrently over a period of time, the system resources (CPU, memory) gradually exhaust, and there are more chances of system failures which couldn't be otherwise identified in sequential execution of the same test scenario.

1.3 Outline of report

1.3.1 ADB Shell and monkey tool

In this chapter a brief explanation of ADB shell and it's working.Then monkey tool is explained with objective,performance and cost,working of monkey.

1.3.2 Testing

In this chapter, Testing is explained elaborately. It also discusses the types of testing such as "manual testing and automation testing" along with benefits and drawbacks.It also talks about the automation hardwares like servo v2,servo v4 and chamelium.

1.3.3 Autotest

In this chapter mainly autotest and testcase are explained elaborately. It also discusses the types of testcases such as client side and server side. This chapter also includes the methods like initialize(), cleanup(), warmup(),setup() which can be overridden.

1.3.4 Execution of testcase

In this chapter,first it discuss about 2 platforms as HOST and DUT is shown for execution of testcases.Then there are different figures as structure of testcase,structure of debug files,random scenario,concurrent scenario,sequential scenario,execution flow of combined script along with execution status of testcases in the form of pie-chart for better understanding.

1.3.5 Conclusion

In this chapter,conclusion of the project is written along with future task.

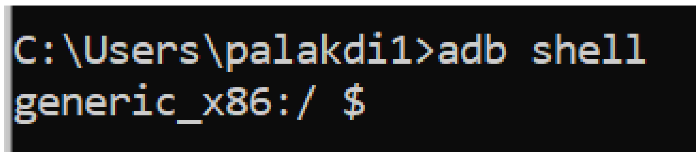
Chapter 2

ADB Shell and monkey tool

In order to explore random scenario from "sequential,random and concurrent test suite",i am using utility of monkey tool and ADB shell of android since monkey tool is concerned with randomness of event.

2.1 ADB

ADB stands for "Android Debug Bridge".ADB is a kind of command-line program which will allows us to interact with tool.ADB includes activities like installing apps and debugging apps.ADB contains 3 parts as client,server and daemon.It is a kind of "client and server" framework.



```
C:\Users\palakdi1>adb shell
generic_x86:/ $
```

Figure 2.1: adb shell

2.2 Monkey Tool

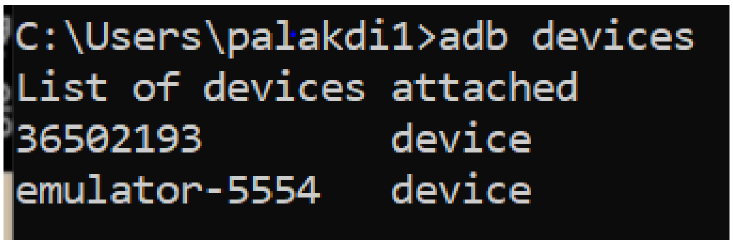
"Monkey is a Program that runs on emulator and which works upon generating the pseudo-random streams of user events such as clicks, touches, gestures. Monkey can runs directly on the emulator by giving adb shell monkey command."

Objective:

The objective of monkey tool is to do Stress Tests as well as Unit Tests. It is helpful to catch frequent bugs and exceptions more fast and more easy. It is also suffiecient enough for handling the system level events in a effective way.

Performance and Cost:
Since the monkey is included as part of the android SDK,it is kept up-to-date by google and has excellent reliability.Running the tool on a virtual device can take a lot of processing power compared to external device.Due to packaged with Android SDK,it is free to use.So,it is compatible with any existing android project.

Working of monkey: For execute the monkey,you must have a device or emulator connected to ADB. At the time of running,you can run only one device at a time,otherwise it will give error.



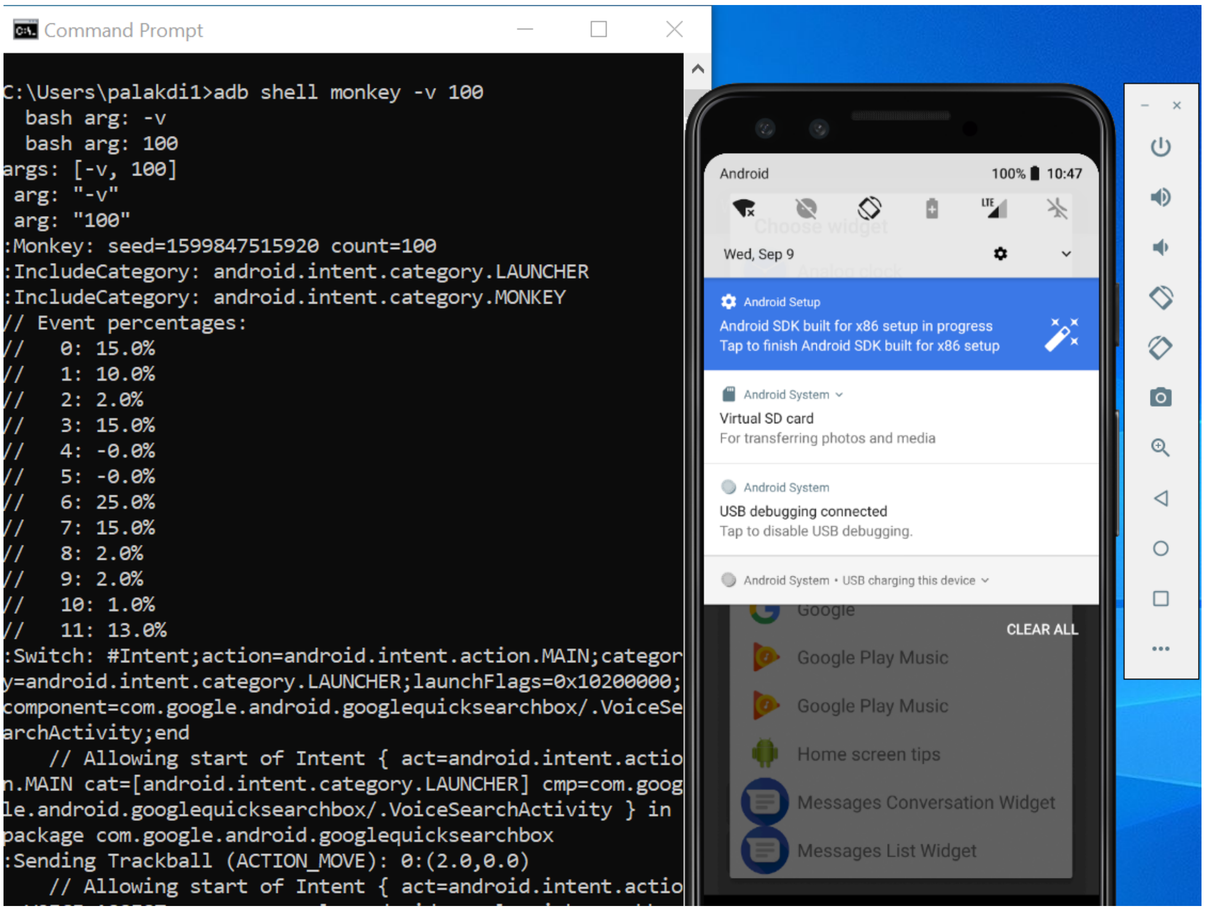
```
C:\Users\palakdi1>adb devices
List of devices attached
36502193        device
emulator-5554   device
```

Figure 2.2: adb devices

2.2.1 Monkey Testing

In monkey tool testing, testing engineer is considered as 'Monkey' by accepting the fact that if any monkey uses the system then it will give some random input without having any knowledge.The testing by monkey tool is a "software testing technique in which the testing is performed on the device under test randomly and it will authenticate whether product application is crashes or not."

Testing engineer provides random data to the application with no predefined test-cases.The objective of Monkey testing is to detect the bugs and errors in the product application utilizing exploratory techniques.Due to this random manner in testing,it is possible that testing engineers can't reproduce those bugs.Monkey testing is done to make sure that the specifications given by the client are properly addressed in the software.Monkey testing is generally carried out as random, automated unit tests.



```
C:\Users\palakdi1>adb shell monkey -v 100
  bash arg: -v
  bash arg: 100
args: [-v, 100]
 arg: "-v"
 arg: "100"
:Monkey: seed=1599847515920 count=100
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
//   0: 15.0%
//   1: 10.0%
//   2: 2.0%
//   3: 15.0%
//   4: -0.0%
//   5: -0.0%
//   6: 25.0%
//   7: 15.0%
//   8: 2.0%
//   9: 2.0%
//   10: 1.0%
//   11: 13.0%
:Switch: #Intent;action=android.intent.action.MAIN;categor
y=android.intent.category.LAUNCHER;launchFlags=0x10200000;
component=com.google.android.googlequicksearchbox/.VoiceSe
archActivity;end
    // Allowing start of Intent { act=android.intent.actio
n.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.goog
le.android.googlequicksearchbox/.VoiceSearchActivity } in
package com.google.android.googlequicksearchbox
:Sending Trackball (ACTION_MOVE): 0:(2.0,0.0)
    // Allowing start of Intent { act=android.intent.actio
```

Android 100% 10:47
Wed, Sep 9

Android Setup
Android SDK built for x86 setup in progress
Tap to finish Android SDK built for x86 setup

Android System
Virtual SD card
For transferring photos and media

Android System
USB debugging connected
Tap to disable USB debugging.

Android System • USB charging this device

Figure 2.3: working of monkey

Monkey Testing is additionally remembered for Android Studio as a feature of the standard testing devices for stress testing.In a few cases, Monkey Testing is committed to "Unit Testing or GUI Testing" too.

Framework of monkey testing:
As per the given below figure, we are able to see that the primary step is to start the DUT. DUT executes in an exceedingly separate method, there's a requirement for synchronization with the completion of the DUT's startup. A primary synchronization purpose is that the planning of most windows.

Many systems display a home screen to alert the user to the continued initialization of the device. Usually, the startup screen disappears and, as a result, the actual main window is displayed when the application is ready for use.In order to stop the interaction with the home screen, the start-up procedure should be customized, for example by specifying additional time.

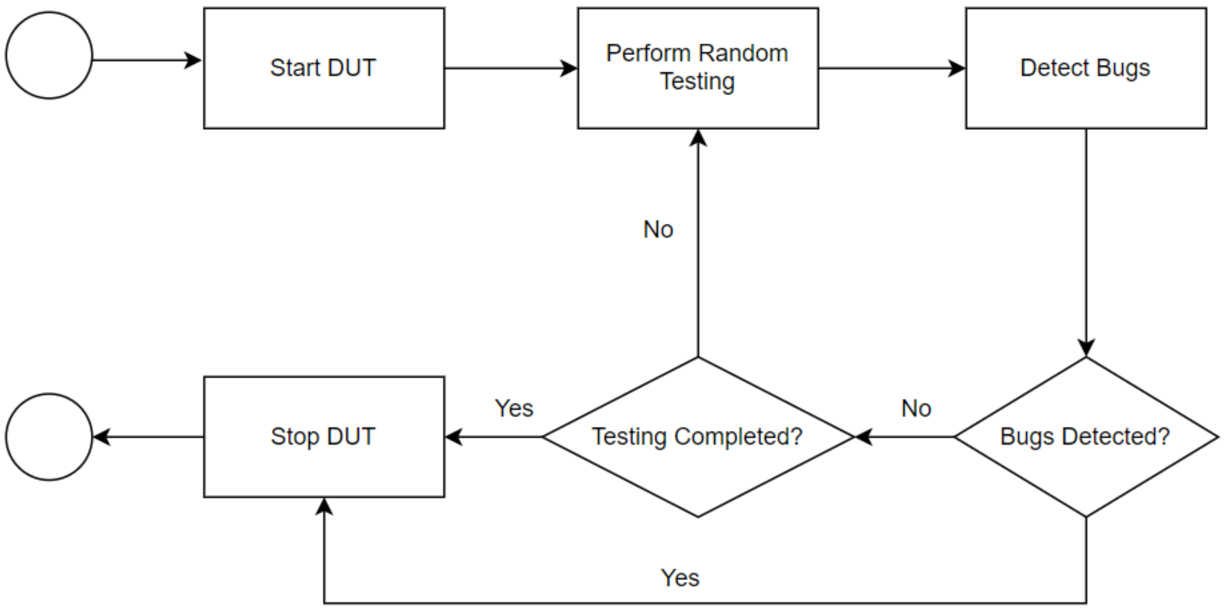


Figure 2.4: Framework of monkey testing

Advantages of monkey testing:

- It is very helpful for testing engineer in the case of full exposure to identify new or say out-of-box errors while executing tests since those are different from previously implemented scenarios.
- It is easy to execute since there is just need of random data to arrange random tests in the system.
- It has benefit in the case of required skills because monkey testing can be performed and accessed by anyone without having proper knowledge of the test or application.
- It is very cost-effective since amount for setting up of environment and execution of testcases is very less.
- It is useful while doing stress testing and load testing as there are generally random and adhoc scenarios for testing.
- Automation of monkey testing is easy with tools.

- It is sufficient for any mobile applications,web applications as well as desktop applications since it's setup is easy.
- it is efficient enough to save the system from time crunch as well as complete breakdown by detecting major bugs and by doing quick fixing for the same.

Chapter 3

Testing

Testing is one of the principal element in the domain of quality assurance in software.Testing can be categorized in 2 ways as automatic and manual.Manual testing is deal with creating a test case in manual manner and execute those test cases with none tool dependency.It includes surfing various application scenarios ,combinations of inputs, compatibility of actual results and expected performance along with keeping track of those observations.At other end,automation testing is deal with executing a testcase using an automation tools and hardwares like servo and chamelium.Objective of the same is to decrease the number of experiments to be run physically and not take out manual testing all together.[11] Testing include running the test,detecting a bug,checking the type of bug by classifying it,developer has to check it and fix that bug and then as a last step new release will be issued to test team.

3.1 Mode of testing

Based on the way of execution of testcases, testing can be categorized in two categories: 1. Manual Testing 2. Automation testing

3.1.1 Manual Testing

Manual Testing is concerned with ”testing method in which the engineers develop their testcase manually as well as executing those testcases in order to find bugs within the product”.Manual testing could be a arduous activity because for that testing engineer should contain specific skillset along with creativity,patience,good observation power,innovative ideas etc. [4][6].Manual testing can be categorized into Integration

Testing,Acceptance Testing,Black Box Testing,White Box Testing,Unit Testing,System Testing.Manually Testing is beneficial to use instead of automation testing when scenarios are like if project is running in it's initial phase,if there is a case when any testcase is unautomatable as captcha, if duration of project is short etc.

Drawbacks of manual testing are following:

- Accuracy is less since it is human-driven because there are chances that testing engineers may commit mistakes
- These testcases takes up huge amount of time since it's executed by human resources and implies a high cost.So it is very time consuming,resources consuming and very slow
- Reliability is less as there are possibilities to get incorrect results since manual verification is prone to human errors in some cases
- Not efficient enough for long term projects and large-scale projects
- Large amount of human resources required as so many testers have to execute testcases manually

3.1.2 Automation Testing

"Automated testing is a teting for any large, long-lived software project to maintain stability while permitting rapid development and Test design and development together can be automated to reduce human effort and save cost "[12].

Benefits of automation testing

- Fast: It is faster than the manual testing.
- Cost Effective: Test cases are executed by using automation tool so less tester are required in automation
- Repeatable: The same test case (record and replay) can be re-executed using testing tools testing
- Reusable: Test suits can be re-used on different versions of the software.

- Programmable: It is programmable by testing engineers.
- More reliable: Automation tests perform precisely same operation each time they are run.
- Test Coverage: Wider test coverage of application features

3.2 Automation hardwares

3.2.1 Servo v2:

- Debug board used in Chrome OS test and development which enables automated testing.
- Provides software access to device GPIOs, EC and UART ports
- Connected to the debug header on the Chrome OS device and to an Ubuntu host machine
- Helps in flashing EC and firmware on the Chrome OS devices
- No longer manufactured but still used in early bring ups

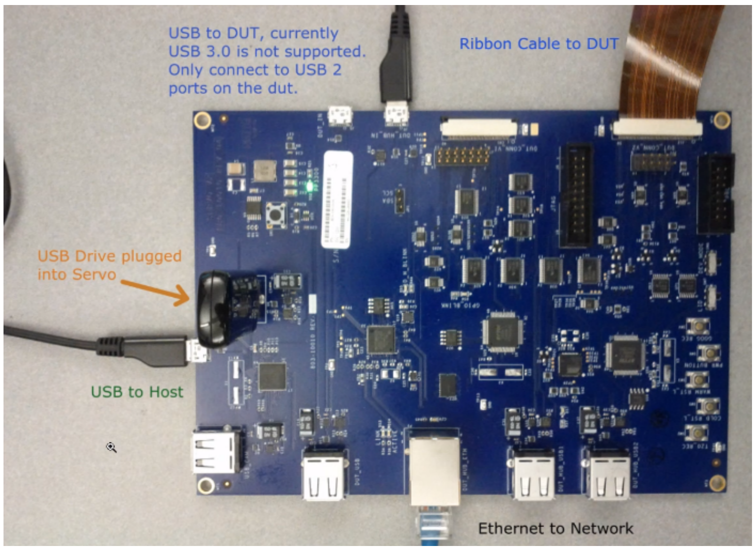

USB to DUT, currently USB 3.0 is not supported. Only connect to USB 2 ports on the dut.
Ribbon Cable to DUT
USB Drive plugged into Servo
USB to Host
Ethernet to Network

Figure 3.1: servo v2

3.2.2 Servo v4:

- Latest version of Servo which is used in newer platforms
- Combines the functionality of the following devices into one:
  Ethernet-USB dongle Muxable USB port uSD to USB 3.0 dongle Pass through charger Case-Closed Debug(CCD) interface

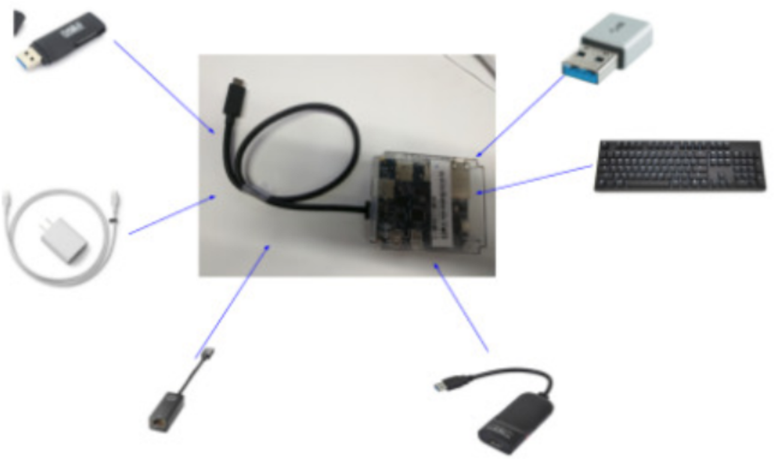
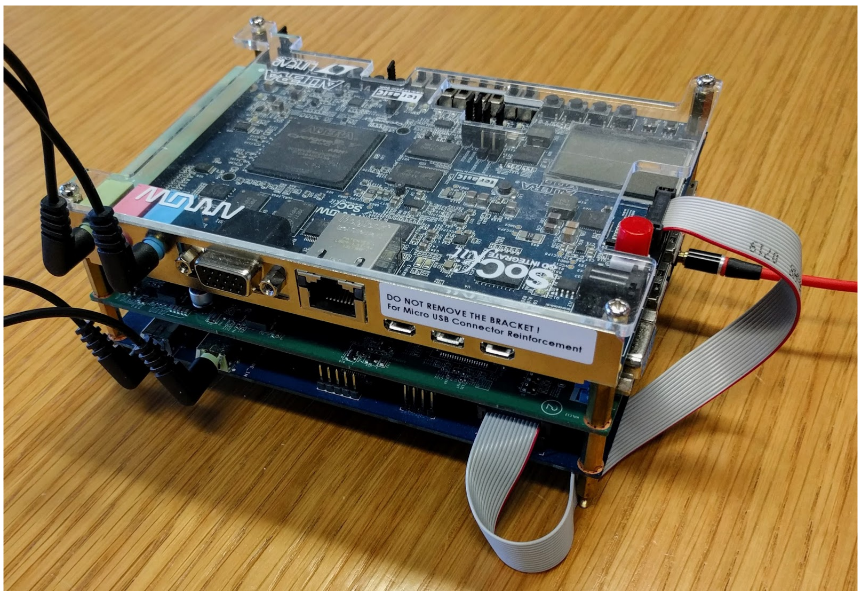


Figure 3.2: servo v4

3.2.3 Chamelium:

- It allows us to simulate all types of displays using EDIDs of the displays.
- It emulates user behavior such as plugging(or unplugging) in an external monitor
- Chamelium receives video and audio output from Chromebooks, converts to RGB(Video) / I2S(Audio) formats and stores in RAM
- Source code for the underlying test infrastructure and logic comes with the Chromium OS source code.


DO NOT REMOVE THE BRACKET !
For Micro USB Connector Reinforcement
Figure 3.3: Chamelium

Chapter 4

Autotest

Autotest is an open-source framework which is working based on doing fully automated testing for large scale systems and low level frameworks like hardware and kernel.By autotest,detection of uncommon defects and unobtrusive execution regression is possible.With as minimal manual setup as possible,it is intended to give start to finish automation for functional tests against executing hardware or kernels.This framework is generally dependent on python.Objective of automation is to diminish the overhead imposed on test engineers or manually execution of testcase as well as not dispose of manual testing at all.Autotest includes activities like handling the erros in implicit manner,generating the reliable outputs,executes either independent or inside any server bridle.In autotest,development of a testcase must be a straightforward,known process,needed of interaction with subset of the available framework[5].

Autotest has some fundamental prerequisites like testcase's outcomes must be accessible in a proper way in which it should be parseable by machine as well as testcases which are developed outside of the system should be able to execute inside the system as well without any problem.

Flow of autotest system:

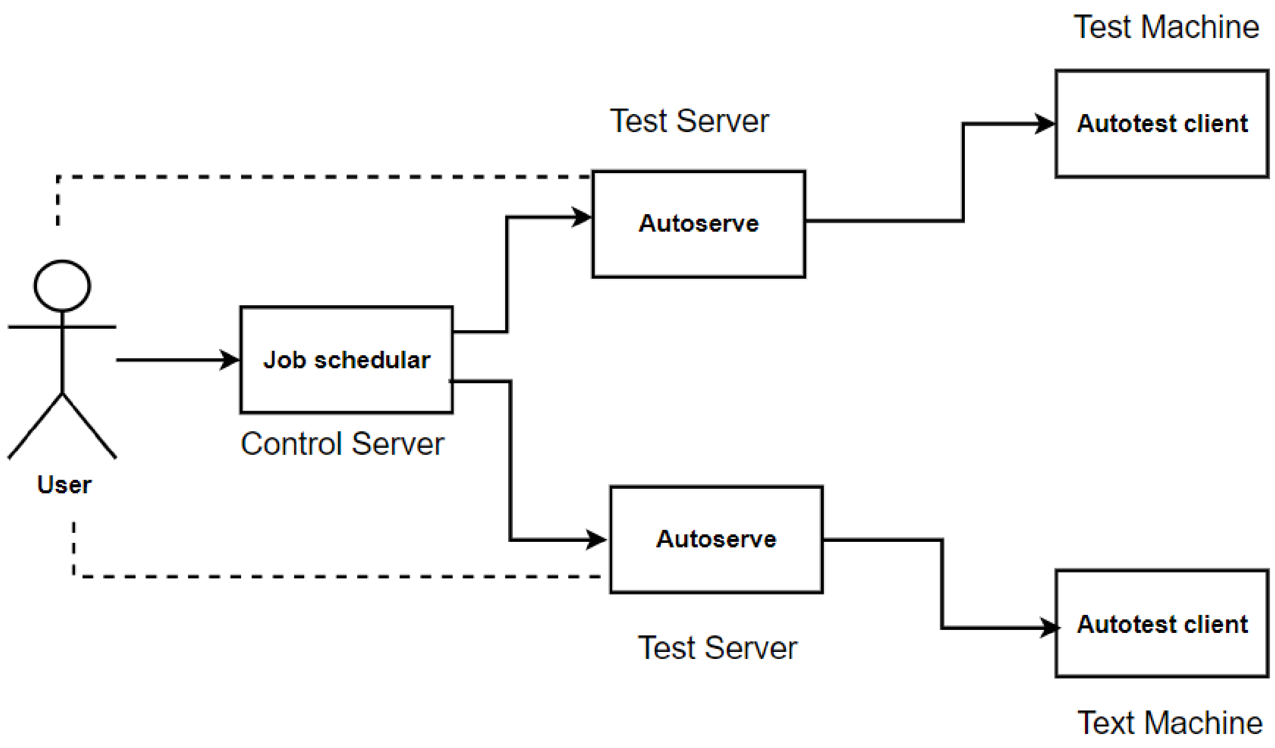


Figure 4.1: Flow of autotest system

4.1 Testcase

The test case is a document in detail which contains input, steps to follow, specifications of a test, expected results and preconditions of testing for a test which is executing on Device Under Test(DUT). Expected result of test may consist output which will come by giving inputs which are selected and it may contains post conditions.A unique test_id which is identification number of that test is there in every testcase.There should be 2 test cases for generating the desired inclusion: one is positive or successful testcase and the other one is negative or unsuccessful testcase.We can say positive testcase when expected result is same as observed result and on the other hand,it is negative testcase when expected result is different from observed result.Every testcase is not able to automate and the selected testcases are the foundation for selection of automation tools and execution of testcases.There is a provision in autotest as ”Suites” which is set of multiple testcases from same domain. At the end,pre-condition should be satisfied with known input and post-condition must be fulfilled by expected result of testcase.

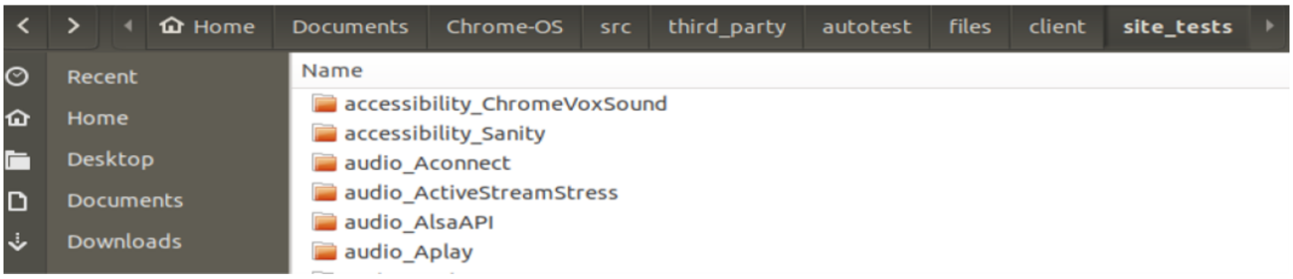
Types of Test cases:

1. Client side

2. Server side

4.1.1 Client side

In client side,Testcases are runs on the DUT.In client side,the whole testcase and related files will be pushed into the DUT and execution will be happening in DUT.Once the execution gets completed,it will give the final result along with logs to the host machine.



accessibility_ChromeVoxSound
accessibility_Sanity
audio_Aconnect
audio_ActiveStreamStress
audio_AlsaAPI
audio_Aplay

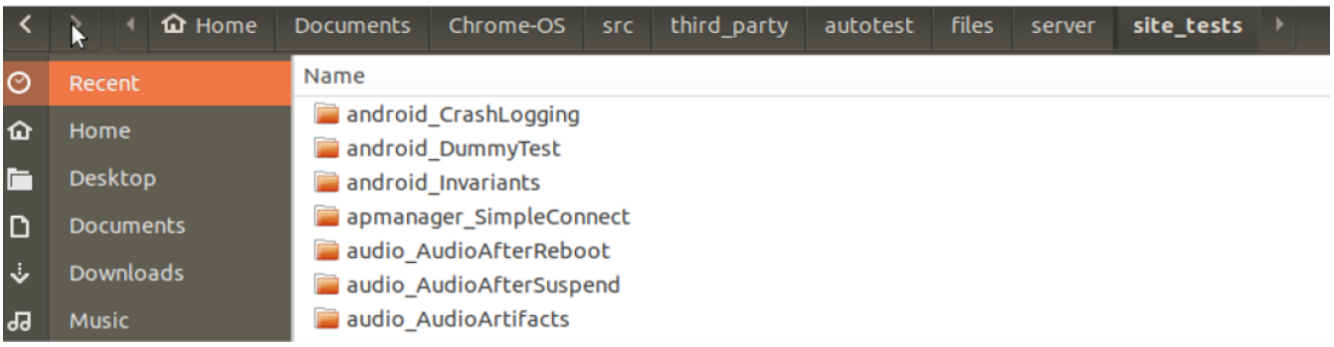
Figure 4.2: Directory Structure of client side testcases

One disadvantage of client side TC is it is not able to maintain states across reboot/cold boot or suspend/resume.As an example,there is 15 lines of code and somewhere in the middle ,if there is any reboot,resume or suspend scenario then it will execute properly that part which is before that scenario and once the DUT comes up,whatever testcase is executing,it will get killed due to this failure scenario.so,even execution is starting again,it will not be able to continue from the line after the reboot,resume or suspend scenario.it will get restart again from the beginning.Due to this reason,server side TC is recommended compare to client side TC.If there are no any reboot,resume or suspend scenario and if there is no need of any automation hardwares like servo and chamelium,then one should start with client side execution.

A client side testcases executes completely based on host machine.Basically the client subdirectory of Autotest is introduced on the host machine toward the start of the test. Thus the control file of client side testcase can execute the script which is in main test class or pythoon file through the job.run_test() fucntion .The location of this test class is in client/site_tests as we can see in figure given here.Test class or autotest_lib.client.bin.test.test is super class of any test class.so,every time we have to pass parameter as test.test while defining a class in main test class since we are extending it in our class.after extending test.test,we have to give run_once() function in our class since run_once is a main runner of any class.In run_once(),we can pass any specific parameters which is required to execute the testcase.In any class,variable version of that class should be there.

4.1.2 Server side

Unlike client side testacases,autotest client is not there on the host machine in server side testcases.server represents an ssh connection by host objects to the host machine.With the help of these host objects,server is able to run the script on client side.In server side,it follows the master-slave kind of architecture where the master will be the host machine and client will be the DUT.In At the time of execution,each and every line of testcase will be picked up and it will be sync with DUT.Execution will happen line by line in DUT and then after that execution,the control will be passed to the host machine.



android_CrashLogging
android_DummyTest
android_Invariants
apmanager_SimpleConnect
audio_AudioAfterReboot
audio_AudioAfterSuspend
audio_AudioArtifacts

Figure 4.3: Directory Structure of server

There are some methods like initialize(), cleanup(), warmup(),setup() etc. which can be overridden in our test class in which initialize() is the first method which is called prior to execution of each testcase,warmup() deals with preparing a pre-test for getting substantial outcome and order of this method is before calling run_once and after doing all setup by setup() method.Then setup() method is called at the time of compilation of script as well as when version of test will be change. cleanup() always called after completion of execution of that testcase in both pass or fail cases as there is possibility that testcase used a large portion of the disk while creating the test files and it will arise an issue with subsequent tests.

The main difference between client side and server side testcases is the point of execution where the testcases are done.Execution time is lesser in the client side TC compare to server side TC.

Chapter 5

Execution of testcase

5.1 Execution for individual testcase

There are 2 facilities in the case of platforms for executing the testcase.

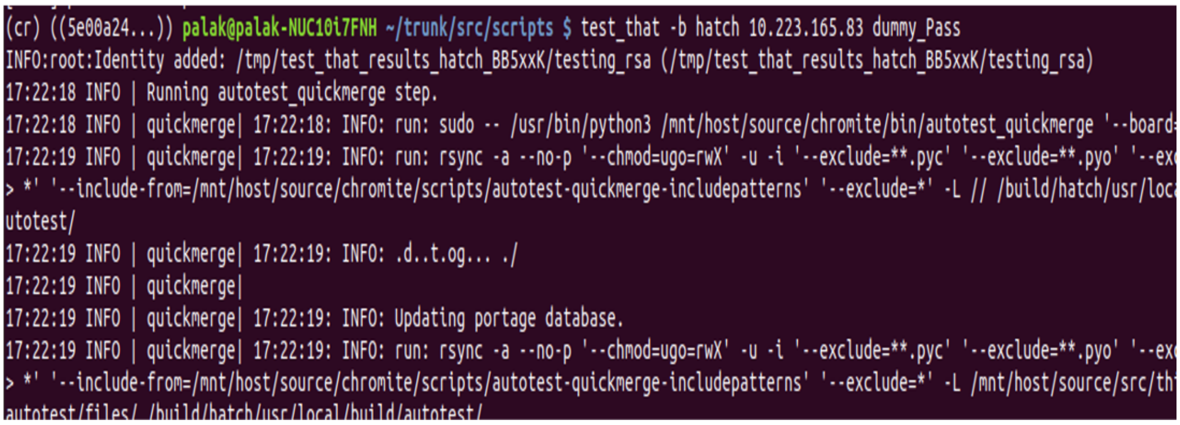
1.On Host machine via chroot

2.On DUT

1.On Host Machine:

Command: **test_that -b $board $host testcase_name**

Here,test_that is main command which we use to execute autotest related testcases.$board is the type of board on which we are executing.Then $host variable deals with DUT_ip followed by testcase name which we want to execute.In given figure,we can see that board is hatch,dut_ip is 10.223.165.83 and testcase_name is dummy_Pass.

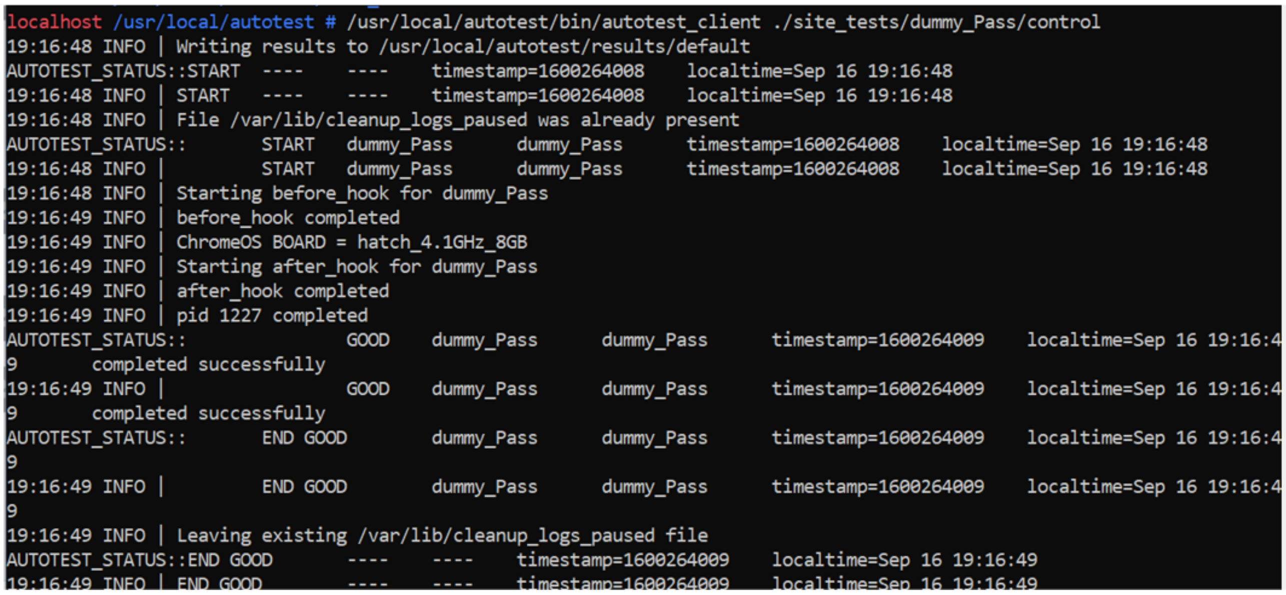


```
(cr) ((5e00a24...)) palak@palak-NUC10i7FNH ~/trunk/src/scripts $ test_that -b hatch 10.223.165.83 dummy_Pass
INFO:root:Identity added: /tmp/test_that_results_hatch_BB5xxK/testing_rsa (/tmp/test_that_results_hatch_BB5xxK/testing_rsa)
17:22:18 INFO | Running autotest_quickmerge step.
17:22:18 INFO | quickmerge| 17:22:18: INFO: run: sudo -- /usr/bin/python3 /mnt/host/source/chromite/bin/autotest_quickmerge '--board
17:22:19 INFO | quickmerge| 17:22:19: INFO: run: rsync -a --no-p '--chmod=ugo=rwX' -u -i '--exclude=**.pyc' '--exclude=**.pyo' '--ex
> *' '--include-from=/mnt/host/source/chromite/scripts/autotest-quickmerge-includepatterns' '--exclude=*' -L // /build/hatch/usr/loc
utotest/
17:22:19 INFO | quickmerge| 17:22:19: INFO: .d..t.og... ./
17:22:19 INFO | quickmerge|
17:22:19 INFO | quickmerge| 17:22:19: INFO: Updating portage database.
17:22:19 INFO | quickmerge| 17:22:19: INFO: run: rsync -a --no-p '--chmod=ugo=rwX' -u -i '--exclude=**.pyc' '--exclude=**.pyo' '--ex
> *' '--include-from=/mnt/host/source/chromite/scripts/autotest-quickmerge-includepatterns' '--exclude=*' -L /mnt/host/source/src/th
autotest/files/ /build/hatch/usr/local/build/autotest/
```

Figure 5.1: Execution on Host machine

2.On DUT:

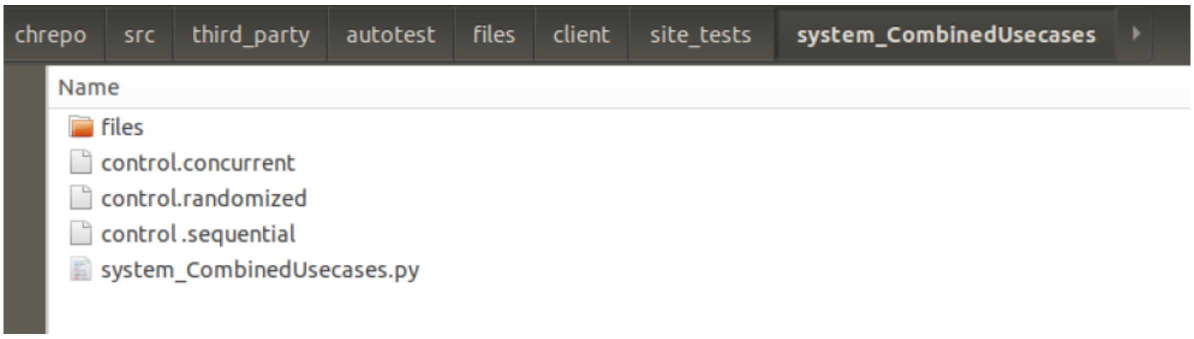
Command: **/usr/local/autotest/bin/autotest_client ./site_tests/test_name/control**
First there is need of copy the testcase to directory "/usr/local/autotest/site_tests".This is the other way of executing the testcase in standalone way.Here autotest_client is a binary file which we have to invoke.This approach is useful when we have to execute client-side testcase as well as when there is unavailability of host machine.



```
localhost /usr/local/autotest # /usr/local/autotest/bin/autotest_client ./site_tests/dummy_Pass/control
19:16:48 INFO | Writing results to /usr/local/autotest/results/default
AUTOTEST_STATUS::START  ----    ----    timestamp=1600264008    localtime=Sep 16 19:16:48
19:16:48 INFO | START   ----    ----    timestamp=1600264008    localtime=Sep 16 19:16:48
19:16:48 INFO | File /var/lib/cleanup_logs_paused was already present
AUTOTEST_STATUS::       START   dummy_Pass      dummy_Pass      timestamp=1600264008    localtime=Sep 16 19:16:48
19:16:48 INFO |         START   dummy_Pass      dummy_Pass      timestamp=1600264008    localtime=Sep 16 19:16:48
19:16:48 INFO | Starting before_hook for dummy_Pass
19:16:49 INFO | before_hook completed
19:16:49 INFO | ChromeOS BOARD = hatch_4.1GHz_8GB
19:16:49 INFO | Starting after_hook for dummy_Pass
19:16:49 INFO | after_hook completed
19:16:49 INFO | pid 1227 completed
AUTOTEST_STATUS::               GOOD    dummy_Pass      dummy_Pass      timestamp=1600264009    localtime=Sep 16 19:16:4
9       completed successfully
19:16:49 INFO |                 GOOD    dummy_Pass      dummy_Pass      timestamp=1600264009    localtime=Sep 16 19:16:4
9       completed successfully
AUTOTEST_STATUS::       END GOOD        dummy_Pass      dummy_Pass      timestamp=1600264009    localtime=Sep 16 19:16:4
9
19:16:49 INFO |         END GOOD        dummy_Pass      dummy_Pass      timestamp=1600264009    localtime=Sep 16 19:16:4
9
19:16:49 INFO | Leaving existing /var/lib/cleanup_logs_paused file
AUTOTEST_STATUS::END GOOD       ----    ----    timestamp=1600264009    localtime=Sep 16 19:16:49
19:16:49 INFO | END GOOD        ----    ----    timestamp=1600264009    localtime=Sep 16 19:16:49
```

Figure 5.2: Execution on DUT

Directory Structure of Testcase: Generally any testcase includes one python file along with required control files as we can see in figure.we can execute testcase with control file which we want to run like an example if there is need to execute testcase in sequential manner then we have to give testcase name as system_CombinedUsecases.sequential while executing the test_that command in host machine.



files
control.concurrent
control.randomized
control .sequential
system_CombinedUsecases.py

Figure 5.3: Structure of testcase

Execution flow of individual testcase:

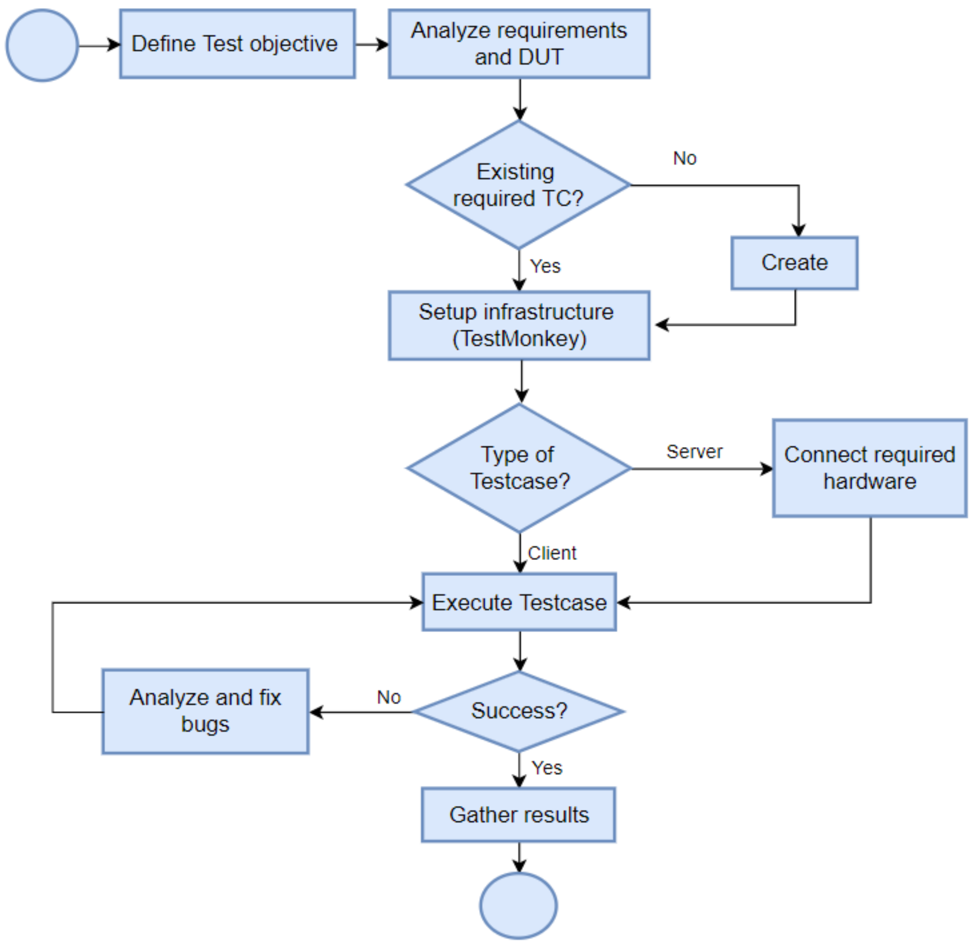


Figure 5.4: Execution flow for individual testcase

5.2 Control Files

The important characterizing part of any testcase is that testcase's control file.This control file is just a Python script which is following autotest python style and which is straightforwardly doing execution of the testcases.Any testcase is defined by it's control file.In control file,there is no any logic part of the testcase. It contains following parameters:

- **AUTHOR** which contains information of the person who had written that testcase
- **DEPENDENCIES** which contains the requirements for executing the testcase

- **DOC** includes arguments to be passed along with description of testcase
- **TIME** defines the duration of execution for the testcase.It can be consider as LONG if testcase is taking more than 4 hours to run the testcase,it can be consider as SHORT if execution time of testcase is less than 15 minutes and it can be consider as MEDIUM in the case when it is taking more than 15 minutes and less than 4 hours
- **TEST_TYPE** parameter says which type of testcase is executing, it can be of client type or server side
- **TEST_CATEGORY** defines the category of testcase which can be stress,functional etc.
- **NAME** defines the name of the testcase

Among all these parameters,AUTHOR,DOC,NAME,TIME and TEST_TYPE are mandatory ones in all the testcases.Important part of any control file is run_test which can be written as:
job.run_test('dummy_test')
we can pass arguments in run_test if it requires any specific parameters as:
job.run_test('dummy_test',test_duration=5, iterations=2)
In this job.run_test(),First argument is the testcase name which is executing and rest all are arguments in the case when testcase requires any specific parameters.

5.3 Different scenarios for execution of testcases

Here there are 3 scenarios for execution which are following:

1.Sequential

2.Concurrent

3.Random

Execution flow of combined script:

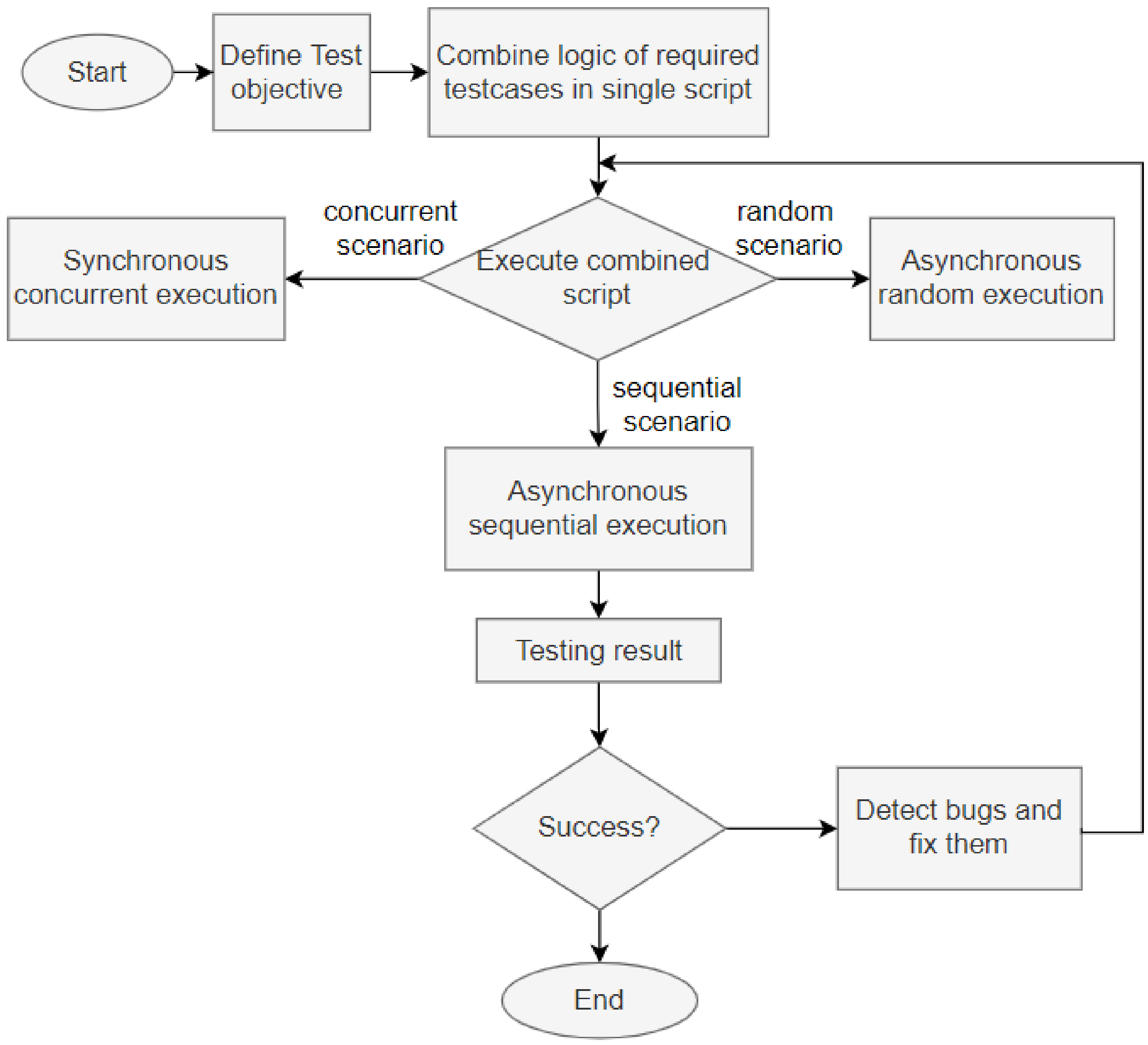


Figure 5.5: execution flow of combined TC

1.Sequential Scenario

In sequential scenario,execution of the functions will happen in order as written in run_once().Those functions will perform one by one.when one function will complete it's execution then only other function will start execution.Since it is stress type testcase as well as it is combining 10 testcases at a time,it is taking much time around 1 hour to execute this combined script.

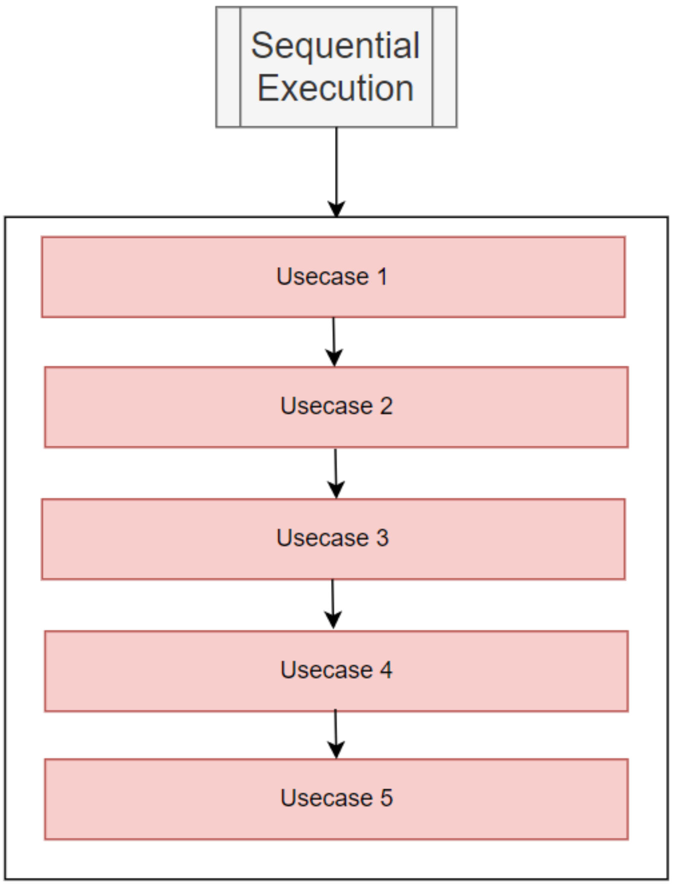


Figure 5.6: Sequential execution

2.Concurrent Scenario

In Concurrent Scenario,execution of the functions will happen in order as written in run_once() but those functions will perform slightly different than sequential Scenario.When one function has started it's execution,after few seconds other function will also start it's execution likewise all the functions which need to get perform,they will perform together at a time in concurrent manner.

3.Random Scenario

In Random scenario,execution of the functions will not happen in the order as written in run_once().The order of execution of those functions will be random.It will pick up any random function from given list and perform it and after completion of that function,it will pick up another any function in random manner.

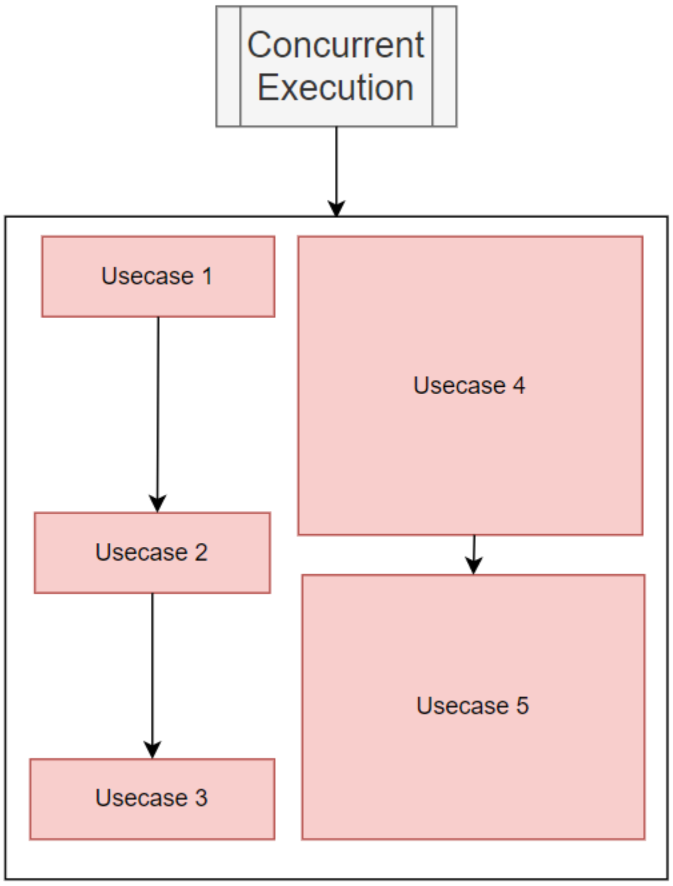


Figure 5.7: Concurrent execution

5.4 Debugging

When testcase is getting failed then there is need to detect a bug,analyze that bug and fix that bug.Every testcase has debug directory through which one can detect the issues occurring during the execution of that testcase.generally debug directory has 4 log levels as DEBUG,ERROR,INFO,WARNING.

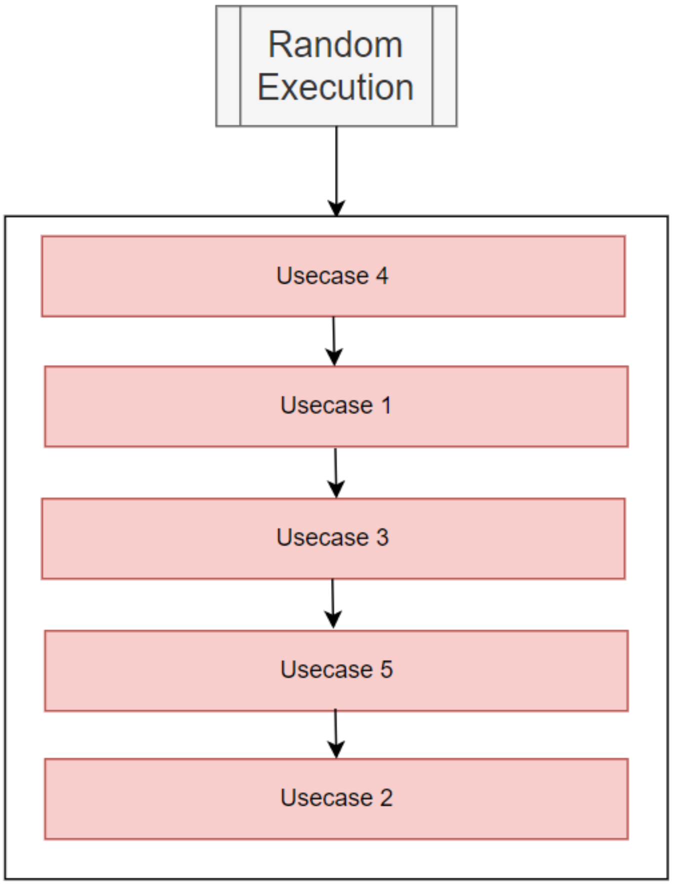


Figure 5.8: Random execution

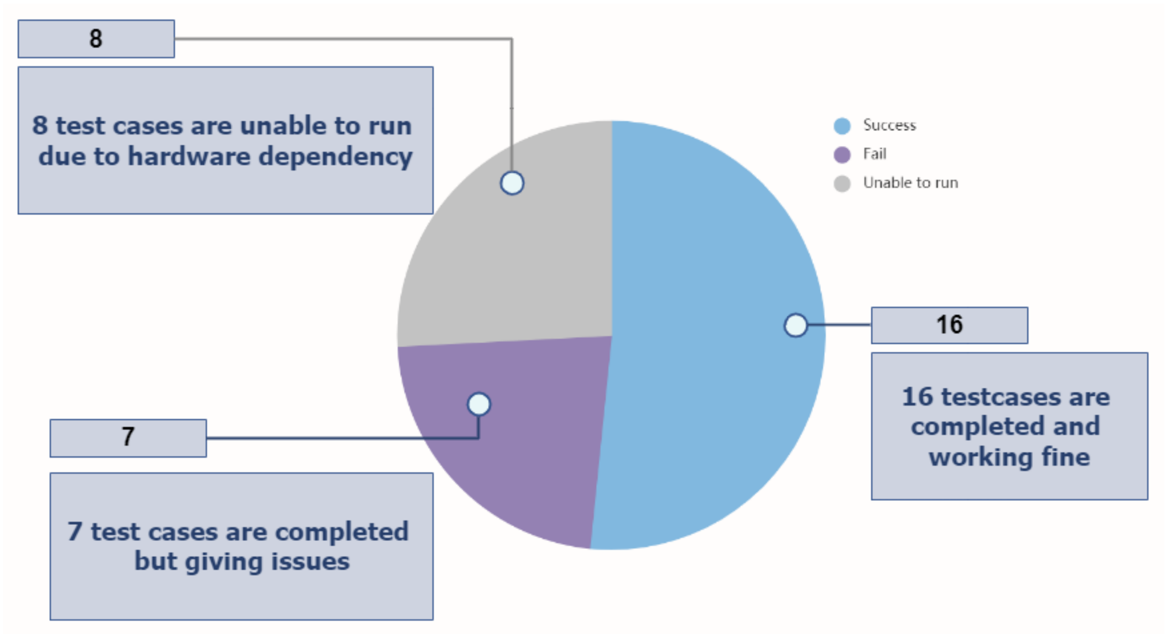


Figure 5.9: Execution status of testcases

Chapter 6

Conclusion

In the first phase of project,detailed study related ADB and monkey tool with the understanding of how monkey testing is done.The monkey tool is fairly simple to use and it is a great for apps that have a significant amount of user input involved. Executing the concurrent tests puts a greater emphasis on user's real-world scenario, and also help to showcase the robustness and performance capabilities.It will ensure stronger validation and an complete automation.

In the second phase of Major project,there is automation related understanding in depth.In this approach, there is in-detail information related to way of executing testcases in automation in all the 3 scenarios as sequential,random and concurrent.

6.1 Future Scope

We are going to give options to users for executing functions as per their wish from the list of functions given in that combined script.As an example,i have 5 functions as app_install(),audio_play(),volume up_down(),brightness up_down() and USB_transfer().Now i want to perform only 2 functions from this list,then at the time of execution it should invoke only these 2 functions instead of all which are written in run_once.

Bibliography

[1] Alzaylaee, M.K., Yerima, S.Y. and Sezer, S., 2017, June. Improving dynamic analysis of android apps using hybrid test input generation. In 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security) (pp. 1-8). IEEE.

[2] Abraham, R. and Erwig, M., 2006, September. AutoTest: A tool for automatic test case generation in spreadsheets. In Visual Languages and Human-Centric Computing (VL/HCC'06) (pp. 43-50). IEEE.

[3] Mouli, C. and Srinivasan, K., 2004, May. Intel automation and its role in process development and high volume manufacturing. In 2004 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (IEEE Cat. No. 04CH37530) (pp. 313-320). IEEE.

[4] Sharma, R.M., 2014. Quantitative analysis of automation and manual testing. International journal of engineering and innovative technology, 4(1).

[5] Admanski, J. and Howard, S., 2009, July. Autotest-testing the untestable. In Proceedings of the Linux Symposium. Citeseer.

[6] Maurya, V.N. and Kumar, R., 2012. Analytical study on manual vs. Automated testing using with simplistic cost model. International Journal of Electronics and Electrical Engineering, 2(1), pp.142-148.

[7] Sharma, R.M., 2014. Quantitative analysis of automation and manual testing. International journal of engineering and innovative technology, 4(1).

[8] Servo,
https://chromium.googlesource.com/chromiumos/third$_party/hdctools/ \quad +$
$/HEAD/docs/servo.mdusing - servo$

[9] Servo v2:
https://chromium.googlesource.com/chromiumos/third$_party/hdctools/ \quad +$
$/HEAD/docs/servo_v2.md$

[10] Chamelium,
https://www.chromium.org/chromium-os/testing/chamelium-audio-board

[11] Testing,
softwaretestingmaterial.com/software-testing/

[12] Automation Testing,
https://www.softwaretestingmaterial.com/manual-testing-vs-automation-testing/

[13] Manual Testing,
https://www.gcreddy.com/2016/04/drawbacks-of-manual-testing.html

Palak Updated file

ORIGINALITY REPORT

7%	8%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Institute of Technology, Nirma University Student Paper	5%
2	www.academicscience.co.in Internet Source	2%

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		