International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:11, 2008

iDEN™ Phones Automated Stress Testing

Wei Hoo Chong

***Abstract*—**System testing is actually done to the entire system against the Functional Requirement Specification and/or the System Requirement Specification. Moreover, it is an investigatory testing phase, where the focus is to have almost a destructive attitude and test not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specifications. In Motorola®, Automated Testing is one of the testing methodologies uses by GSG-iSGT (Global Software Group - iDEN™ Subcriber Group-Test) to increase the testing volume, productivity and reduce test cycle-time in iDEN™ phones testing. Testing is able to produce more robust products before release to the market. In this paper, iHopper is proposed as a tool to perform stress test on iDEN™ phonse. We will discuss the value that automation has brought to iDEN™ Phone testing such as improving software quality in the iDEN™ phone together with some metrics. We will also look into the advantages of the proposed system and some discussion of the future work as well.

***Keywords*—**Testing, automated testing, stress testing, software quality.

I. Introduction

TESTING is an important phase in the software development cycle. It can be done by either manual testing or automation testing. The testing scope and activity is designed in feature test suite basis [1]. Software testing is the activity of running a series of dynamic executions of software programs that occurs after the software source code has been developed. It is performed to uncover and correct as many of the potential errors as possible before the software is delivered to the customer [2].

In GSG-iSGT, we are practicing manual and automated testing with the purpose to produce good quality product, which in turn will increase the customer satisfaction. For instance, there is no customer would want a phone that resets or hangs just because he made a few continuous calls, and etc. The improvement in customer satisfaction equals to the increase in the trust of the organization's products. In the long term, this would bring an increase in the market share and with that profitability. In order to achieve this goal, one of the methods is stress the phone to its limit and the phones still maintain its stability.

This paper is thus presented as following order. The next section browses through some related works, and then an idea of iHopper is explained as well including its modules and also how it operates in a test environment. Next, we will see how iHopper can produce more robust product, reduce the cycle time compare to conventional manual testing, and increase the productivity by supported with some metrics. This will then followed by the advantages and disadvantage of iHopper as an automated testing tool. Finally, there is some discussion on the future work and this paper is ended with conclusion.

II. Related Works

Recently, there are a lot of automated testing tools that Motorola® GSG-iSGT team using in System Testing process, for instance, iPTF (iDEN Phone Test Framework) [3] [4], iRobot [5], and etc.

A. iPTF (iDEN™ Phone Test Framework)

iPTF [3] [4] is a standalone application, which is currently the standard test tool used by the Motorola® test team to execute test scripts locally and on remote site where all the iDEN™ phones or device-under-test located. Please refer to Fig. 1 for the existing framework of iPTF. If more than one iPTF sessions are needed, the Motorola® test team will configure every session with the appropriate settings at different machine. We can run more than one iPTF session in the same machine, however, this will cause the machine been overloaded and decreasing the overall test productivity.

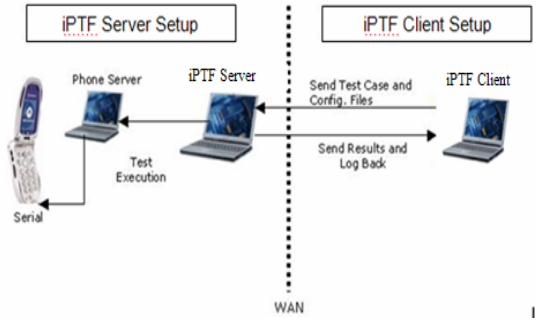

Fig. 1 iPTF Architecture [3]

For the remote execution case, all the test scripts, developed in a local site is then compiled using a JAVA™ compiler, and configuration items such as test environment, needed to be done locally prior to the test execution. Then the iPTF is sending every command from the test scripts from the local site to the remote site to be executed. After execution is completed, the results are sent back to the local iPTF for evaluation via network. This method consumes a lot of bandwidth as each command is sent via the network and results returned via the network in real time.

B. iRobot

iRobot [5] serves as a tool to perform stress test on iDEN™ Gemini Smartphone. The basic concept is that users can

manipulate the Smartphone to replay recorded scripts for multiple numbers of times at some future date to stress the phone to its limits. They will also be able to playback more than one script in a specific pattern and at a specific number of times, again, to stress the phone. Both the iRobot scripts and the pattern files can be transferred and replayed on another Smartphone. Besides recorded scripts, iRobot also has built-in stress tests such as data call, phone call, and random test. These embedded tests serve to allow more thorough stress testing on the different areas of the Smartphone.

Fig. 2 iDEN™ Gemini Smartphone

iRobot was used by a team in India and Malaysia to stress test Gemini phones. Fig. 2 is iDEN™ Gemini Smartphone.

Record and Playback feature allows user executing testing many times by only creating test cases once. Besides, user is able to reproduce the same steps of testing that was executed previously. Moreover, random keys pressing are able to stress the phone to its limit. With this information, we are able to design and produce more robust and reliable phones in order to meet customer satisfaction. However, there is one main issue with iRobot, which is iRobot only support mobile windows platform. So, in this paper, we would like to introduce another automated system testing tool called iHopper to support other than mobile windows platform, which is iDEN™ UIS platform.

III. iHOPPER

iHopper [6] is a stress testing tool for iDEN™ phones, which is having similar features as iRobot and RPB. It is based on the same concept as the Microsoft provided Hopper tool for Gemini. It provides random keystrokes to the phone under test. The concept is that the phone is driven randomly through the UI, and therefore stresses the phone software.

iHopper differs from Hopper in the following ways:

- Only runs on iDEN™ UIS phones (not on WinCE phones).
- It is an application that runs on a host PC (Personal Computer), rather than on the SU (Subscriber Unit) itself.
- Logs are stored on the host PC.
- The tool uses the iPTF test agent, so this will work on both Lab builds and MS builds with no further installation or configuration.
- Key inputs are random, but the probability of a given key being pressed varies. An algorithm makes the probability of a given key vary as a function of the previous keys. The goal is to reduce the probability of events that would terminate the sequence of entering menus (such a flip events or pressing the home key), and to enhance the probability of entering applications (by making the OK key, and up/down keys more likely under certain circumstances.) Thus, randomness is preserved, yet the likelihood of the phone navigating deeper into menus is enhanced.

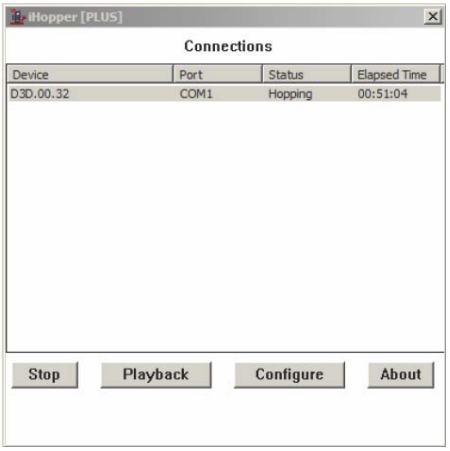

Fig. 3 iHopper Tool View

Fig. 3 is shown the iHopper user interface. Once a device-under-test (iDEN™ Phones) is connected to the iHopper, the interface lists the software ID/version of this device, the port that the device is connected to, the status of the test, and the elapsed time that the test has been running. The elapsed time updates about every 10 seconds. Multiple devices can be connected. Each device will have a corresponding status line on the main interface.

Once a system has been configured, phones are connected, and iHopper is running, no further user interaction is required. If desired, the user can terminate all iHopper sessions by exiting this main interface.

IV. iHOPPER MODULES

There are 4 main modules in iHopper.

A. Playback

Playback module allows user to reproduce/ playback the same random sequence of key presses that was executed previously. There are two methods to do this.

1. Random Seek

User can specify a random seed of a given log file to reproduce the same sequence of key presses. Fig. 4 is an interface for user to enter the random seed. Every new random seed is generated for every session of execution iHopper and is logged in the log file as shown in the sample iHopper logs in Fig. 5 below.

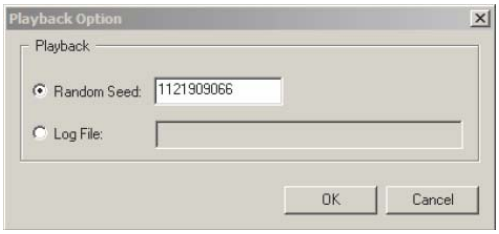

Fig. 4 Random Seed Playback Method

```
09:24:26:268 - iHopper Version: iHopper Plus (00.05)
09:24:26:268 - Random Number Seed: 1121909066
09:24:26:268 - New session started.
09:24:26:268 - Version Number of Device Connected: D5B.00.06.
09:24:26:268 - Time Connected: 20050721 092426.
```

Fig. 5 Sample Log with Random Seed number

If a user wishes to reproduce the same random sequence of a specified random execution, they can do so by providing a random seed number of that execution in this field. Once specified, a new session will use this random seed when a new connection is made.

2. Log File

Optionally the user can also playback a specific log file by specifying the log file name as shown in the Fig. 6 below. This will reproduce the same key presses as in the log file.

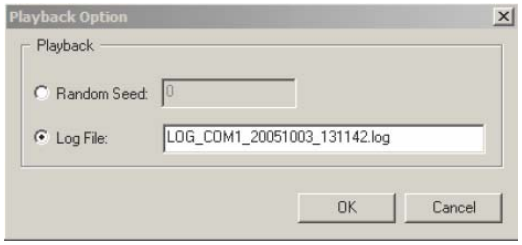

Fig. 6 Log File Playback method

When the log file is loaded, the configuration values logged in the log file will be loaded by iHopper so as to reproduce the same setting when the log file was originally created.

B. Configuration

iHopper is designed in the way that required minimal user interaction. Thus, appropriate information is needed to provide to iHopper so that it can execute correctly. This information includes,

- Option to prevent certain key sequences. By default the following keys are prevented:
  - Pressing Menu * (keypad lock)
  - Pressing keys not available when flip is closed
  - 911, 112, 119 key sequences (emergency calls)

- Option to explicitly prevent keys from occurring more frequently than every specified number of key presses. By default the END key is prevented from occurring more frequently then every 30 key presses.

- Option for user to define key codes to include additional keys or exclude redundant keys.

- Option to turn on/off Flip and specify valid keys when flip is closed.

In addition, there are few configuration items such as Ports (which port is been used), Baud Rates (used to communicate with the phone), Key Interval (delays between key presses), Timeout (the timeout time for iHopper to retry before determining the phone is unresponsive), and Plus options (Plus mode enables deeper navigation into menus) are been configured and stored in the computer after iHopper runs. Thus, these settings need only be adjusted once, or as the configuration is changed, which is aiming to minimize the user interactions as much as possible.

C. Output Log

Output Log is the most important module in iHopper. All the phone activities, test results and system messages during each iHopper session are logged into log file. This log file is then used by test engineer or developer to analyze the root cause if failure happened. Fig. 7, 8 illustrated the example of the log file obtained from iHopper with explanations highlighted in yellow. These logs contain the time that each session started, the specific keys that were input to the phone, as well as keystroke and duration statistics for each session.

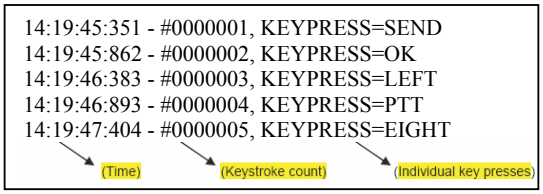

Fig. 7 Key Presses Log

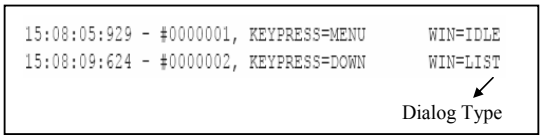

Fig. 8 Key Presses Log in PLUS mode

D. Automatic Ram Dumps

Ram dump is a log file which can be obtained from the phone when the phone is reset / hanged. It contains the phone failure information which may helpful to developer when debugging. Thus, automatic taking ram dumps is necessary in iHopper once a reset is detected on the phone since there is no interaction from a user after the iHopper is begin executed. Once the ram dump file is obtained, iHopper will restart the phone and a new session will start automatically.

Ram dump files will be located at the same location as iHopper logs and will be named the same as the iHopper log that was taken when the reset occurred.

Ram dumps are not possible on USB connected phones. Phones that reset on USB connections will not automatically restart by iHopper. Besides, the phone has to be loaded with Lab loads in order to have automatic ram dumps capability. Otherwise, if MS loads are used, the tool can still be used, but no ram dumps will be taken.

V. Success Story with Results

Fig. 9 illustrates the number of software defects that discovered by using iHopper automated stress testing tool within 9 months on Marlin before the product is release to the market. This data consists of various features tested, several of software release version and it was based on only ten engineers in whole world Motorola® test team with more than 5 iDEN™ phones per engineer. The automated test was executed 24 hours per day without any human intervention. Test engineers only need to configure all the necessary settings in the iDEN™ phones and iHopper before the execution begin. The iHopper is acted like a virtual tester and logs the results to a log file by itself.

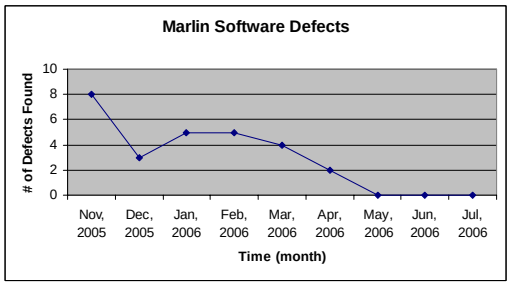

Fig. 9 Number of software defects found by iHopper

From the graph, we noticed that the software defects decreased to zero on May 2006 from 8 defects on Nov 2005. This is a good sign as the defects can be identified before the product is released to the market.

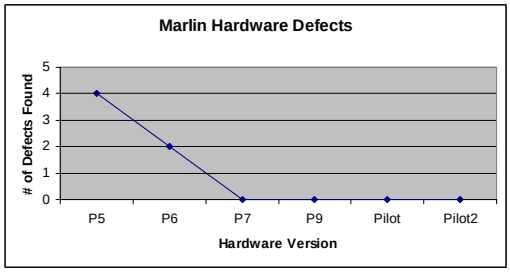

Fig. 10 Number of hardware defects found by iHopper

In other hand, iHopper also able to discover some hardware defects as shown in graph Fig. 10. As we can see from the graph, the early hardware, P5, is not as stable as P7 and later. The quality of the hardware is improved significantly.

In short, iHopper not only will discover software defects but also hardware defects. Utilizing these kinds of automated testing tools in test execution, we are able to design and produce more robust and quality products to customers.

A. Cycle Times Reduction

With the automation helps, the manual testing cycle time is reduced at least 3X to deliver the same quantity of job. How to prove that? We assume that each engineer only works 8 hours per day consider as one Staff day and EACH iHopper will execute for 24 hours per day. As a result, we are able to reduce the cycle times up to 3X. Please refer to the equation (1) below to see how it can be done.

$$\text{Cycle Time} = \frac{(R * I_N)}{S_D} \quad (1)$$

Where
R – Number of hours iHopper executes
I_N – Number of iDEN™ phones used
S_D – One staff day in hours

In this case, within 9 months, the manual testing cycle time is reduced around 150X with 5 iDEN™ phones per engineer.

B. Productivity

In another word means, with the less cycle time will increase the productivity. This is shown by the equation (2) below.

$$\text{Productivity} \ \alpha \ \frac{TC}{\text{Cycle Time}} \quad (2)$$

The productivity is inversely proportional to cycle time spent on testing and proportional to number of test cases assigned, TC.

As iHopper can be executed by itself without any test case and only need some configuration on the phone and iHopper once there is a new phone setting. Besides that, users do not have to keep an eye over each test run. They can check the results in the log file after each execution. Thus they will have the time to focus on other tasks while the stress test is running on the phone.

VI. Advantages of iHopper

The advantages of using iHopper are

- Total cycle time can be reduced. With iHopper, the cycle time can be reduced at least 3X if compared to manual testing and increased the productivity.
- Reusable: The key-presses logs can be used again in the future to reproduce the same sequence of key presses. Eliminating in writing scripts, thus saving the time.
- Reliable: Eliminating human error because tests perform precisely the 'same' operations each time they are run.
- Flexible: No special attention is needed too, and users can leave the scripts running without having to keep an eye over it as the result of each test will be logged in a log file. Users need only to refer back to the log file which contains easy-to-read yet detailed information of every test run.

- Saving Cost: as less resource is needed to operate the test.
- No programming knowledge is required from the users as the tool can be operated easily through a few key-presses.

In short, this tool serves a lot in time yet it allows means to stress the phone to its maximum.

VII. Future Works

iHopper still can be enhanced to have more features, such as having same capability as iPTF which is able to verify the functionality in the released phone software. This feature is able to make entire testing more thoroughly.

VIII. Conclusion

This paper studies automated testing tool, named iHopper, in stressing all the iDEN™ phones except iDEN™ Smartphone. Hence, we are able to determine the limit of our products. Thus we are able to further improve it if the limit is unsatisfactory before the actual release of the product to the customer. Besides, automated testing also able increase the productivity of testing and reduced the manual testing cycle time significantly. In short, the tool enables us to improve on the quality of the phone software and hardware at the same time.

References

[1] Muhammad Aiman Mazlan, Ong Kein Wei, Cindy Phang Sim Sim, iDEN Phone Testing: Automation vs Manual Testing, 3rd Motorola China Technology symposium, 2006.
[2] W H C Bassetti, W. E Lewis, "Software Testing and Continuous Quality Improvement", 2004. pp. 289-290
[3] Wei Hoo Chong, Market-Based Resource Sharing Job Scheduler, *Proceedings of the International Conference on Information Technology and Multimedia,* ICIMU'05, Malaysia, 2005.
[4] Wei Hoo Chong, iDENTM Phoens Automated Testing in P2P, icns'06, ISBN 0-7695-2622-5, 2006.
[5] iDEN Subcriber Test Group, Design Document for iRobot, version 0.0.1, Motorola Malaysia Software Center, 2004
[6] iDEN Subcriber Test Group, iHopper User Guide, version 00.01.00, Motorola Malaysia Software Center, 2005

Wei Hoo Chong presently working as a iDEN™ VRIS engineer in Global Software Group (GSG), Motorola Multimedia Sdn. Bhd., Penang, Malaysia. Previously he was a test engineer for about 2 years and has been awarded as Top Defects Founder of the year 2005. He was a research officer in University Sciences Malaysia before joining Motorola. He completed his MSc. Computer Sciences major in distributed computing and networking within a year from University Sciences Malaysia, Penang, Malaysia in year 2003. His master thesis is on searching in P2P environment by using JXTA. He becomes a Member of IEE in year 2005. He can be reached by hoochong@motorola.com.