Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, **2015**, *7*, 353-357

Open Access

Design of a Python-Based Wireless Network Optimization and Testing System

Hongxue Yang*

Beijing Polytechnic, Beijing, 100176, P.R. China

Abstract: As an object-oriented dynamic scripting language, Python has been widely applied in scientific computation, image processing and network communication fields, etc. However, there is no enough attention paid to it in automated test. A Python-based client/server auto-dial testing mode, which is successfully applied in actual wireless network optimization testing, is introduced. The actual application shows that this mode brings flexible and high-efficiency features of Python programming into full play and significantly improves the efficiency of testing system.

Keywords: Automated testing, Python technology, Wireless network.

1. INTRODUCTION

Developing from 3G to 4G is the trend of communication industry, and large-scale construction of CMCC's TD-LTE base station and other network equipment has already begun. According to the calculations, investment put by CMCC in 4GLTE base station equipment has increased from 4.65 billion yuan in 2012 to 27.5 billion yuan in 2013. With successful development of TD-LTE, the communication technologies which are dominant in China, will be developed and evolved continuously [1]. In addition, the large-scale construction of TD wireless network contributes to the increasing demand on the auto-dial testing software corresponding with it. A method for design and realization of this Python-based wireless network automated testing system is presented in this paper. The application shows that this kind of structure may bring into full play the flexible simple Python programming and high VC implementation efficiency features, improve the testing efficiency markedly and save the manpower cost.

2. CHARACTERISTICS OF PYTHON LANGUAGE

The predominant characteristic of Python lies in rapid development function. Compared with any other programming languages, Python enables software designers to pay more attention to the problem itself, rather than implementation of details. Its main features are listed as follows:

(1) Object-oriented: Python supports both process-oriented and object-oriented programming. In the "process-oriented [2]" language, the program is constructed by either processes or only functions of reusable codes. However, in the "object-oriented" language, the program is constructed by objects which are composed of both data and functions.

(2) Expandability: if you want to run a section of key code faster or keep some algorithms private, write part of programs with C or C++ and then apply them in Python program.

(3) Embeddability: Python can be embedded into a C/C++ program to provide scripting features for program users. Fast speed: running speed is very fast as the underlay of Python and most of standard libraries and the third-party libraries are all written with C programming language.

(4) Transferability: owning to its nature of open sourcing, Python has been transferred to various platforms [3].

(5) Interpretability: inside the computer, Python interpreter converts the source code into an intermediate form of byte code and then interprets and runs it in the form of machine language that can be identified by the computer, which makes Python simpler and easier to be transferred.

In wireless network auto-dial testing system, just like building blocks, the testers can write scripts by means of Python interface functions of open test service, and flexible configuration of test service can be realized with a limited programming knowledge.

3. APPLICATION OF PYTHON TECHNOLOGY IN WIRELESS NETWORK AUTOMATED TESTING SYSTEM

3.1. System Structure

The system consists of two major parts: CMP (Central Management Platform) and MSP (Mobile Server Probe), as shown in Fig. (**1**).

Where, CMP software, mainly used as the client for users, incorporates monitoring and management of terminal status, analyzing, compiling and executing of Python scripts, and centralized display of terminal data and network testing information, etc. As shown in Fig. (**2**), the MSP software takes charge of communication with CMP, real service exe-

*Address correspondence to this author at the College of Telecommunication Engineering, Beijing Polytechnic, Beijing Chaoyang District Road No. 2,100016, P.R. China; Tel: +86 10 13641042169; Fax: +86 10 64316740; E-mail: yhxzxb@sina.com

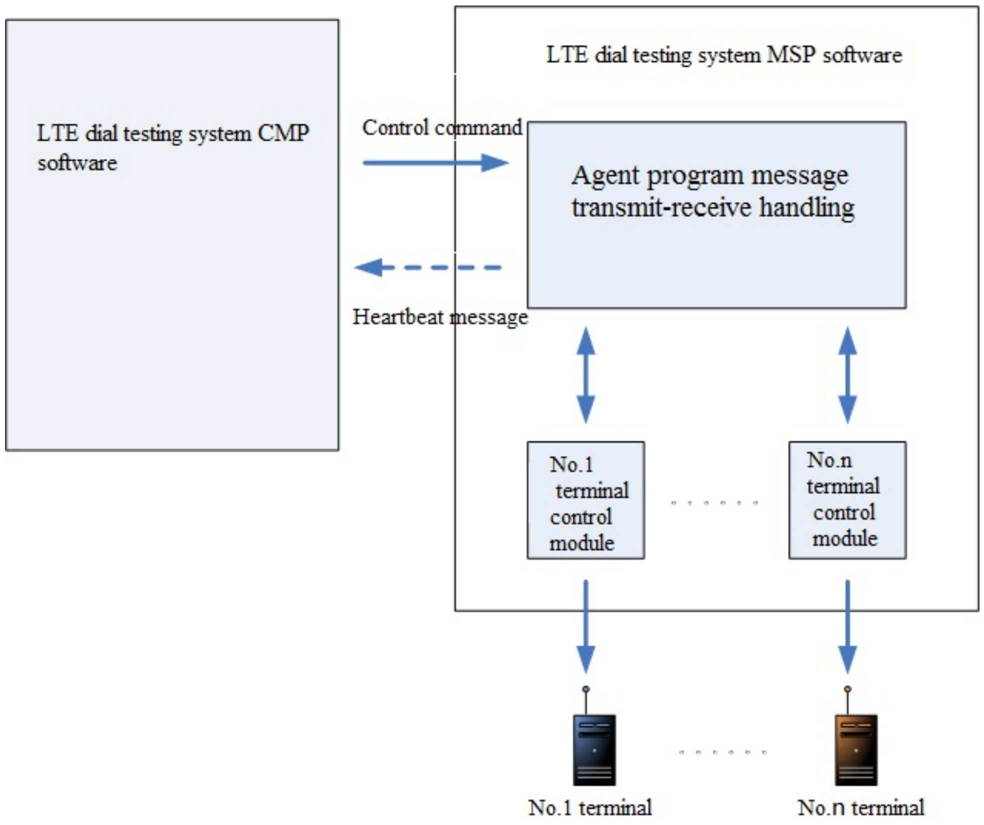
LTE dial testing system CMP software
Control command
Heartbeat message
LTE dial testing system MSP software
Agent program message transmit-receive handling
No.1 terminal control module
No.n terminal control module
No.1 terminal
No.n terminal

Fig. (1). Overall System Framework Diagram.

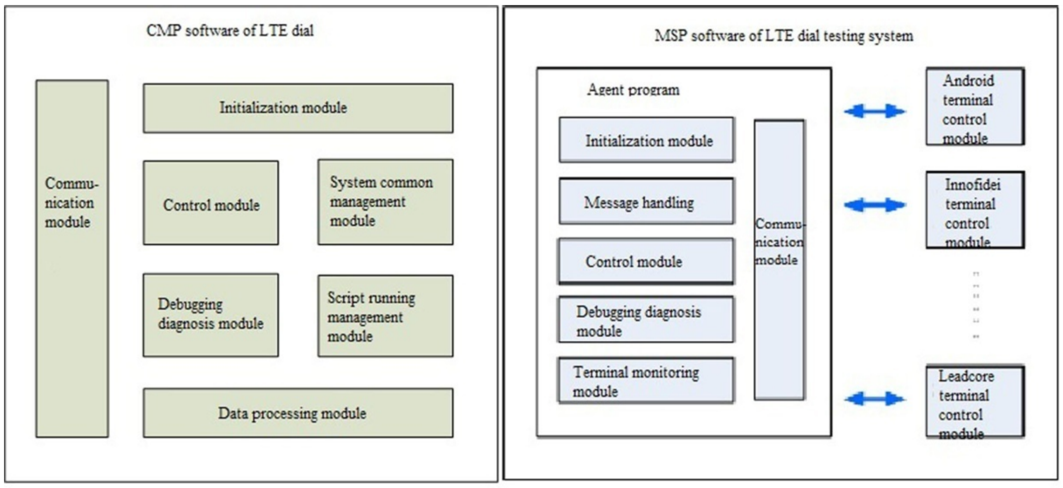
CMP software of LTE dial
Commu-nication module
Initialization module
Control module
System common management module
Debugging diagnosis module
Script running management module
Data processing module
MSP software of LTE dial testing system
Agent program
Initialization module
Message handling
Control module
Debugging diagnosis module
Terminal monitoring module
Commu-nication module
Android terminal control module
Innofidei terminal control module
Leadcore terminal control module

Fig. (2). Function Block Diagram of CMP and MSP.

cution of control terminal, automatic identification of terminals provided by different manufacturers and acquisition of terminal information, etc.

3.2. Design of Python-Based Auto-dial Testing Operation Mode

For purpose of convenient expansion and flexibility in use of auto-dial testing system, Python scripting language is now introduced and serves as the writing and running space for use cases. In order to ensure that testers will not be affected by introduction of Python scripts, client/server operation mode is adopted between the testing script and the auto-dial testing system. The auto-dial testing system serves as a server while the scripts constitute the client. A UDP interface is added in Python operand, in order to execute the function to send corresponding control commands to PythonSERVER; it is necessary to add a PythonSERVER to the auto-dial testing system side and complete corresponding sorting and abstraction of original ATP functions, so as to form an API interface and make preparations for calling of scripts (as shown in Fig. **3**).

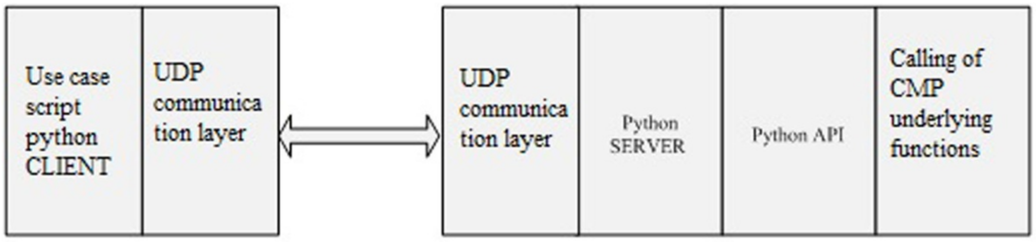

Fig. (3). Operation mode of auto-dial testing system's client/server.

Table 1. Design of auto-dial testing system's Python interface.

Interface Name	Function Description	Parameter Description
MultiAdtOpenCom(Adt_Id，nPort，strType)	Open COM port	[Adt_Id, Id number of MSP], [nPort, port of number], [strType, type of UE]
MultiAdtCloseCom(Adt_Id，nPort)	Close COM port	[Adt_Id, Id number of MSP], [nPort, port number]
MultiAdtDialConnect(Adt_Id，strNumber, strConnectName，nPort，strUeType)	Connect to a network	[Adt_Id, Id number of MSP], [strConnectName, name of network card],[nPort, opened port], [strType, type of UE]
MultiAdtDisConnect(Adt_Id, strDisConnectName，nport，strUeType)	Disconnect from net-work	[Adt_Id, Id number of MSP], [strConnectName, name of network card]
MultiAdtDownLoad(Adt_Id，strIP，strUseId, strPassword，strDownLoadFileName, strInterFaceName，strUeType)	Download service	[Adt_Id，id number of MSP]，[strIP，FTP Server IP] [strUseId，FTP ServerName][strPassword，FTP Server password] [strDownLoadFile-Name，DownLoadFileName] [strInterFaceName，InterFaceName] [strType，UeType]
MultiAdtUpLoad(Adt_Id，strIP，strUseId, strPassword，strUpLoadFileName, strInterFaceName，strUeType)	Upload service	[Adt_Id，id number of MSP][strIP，FTP Server IP][strUseId，FTP ServerName][strPassword，FTP Server password] [strUpLoadFileName，UpLoadFileName] [strInterFaceName，InterFaceName][strType，UeType]
MultiAdtAddRouteMetric(Adt_Id	Add routing table	[Adt_Id, id number of MSP]
ATPCommand(CommandName，*Parameter)	General client interface	[CommandName, name of commands][Parameter, parameters]
SendData(strData)	Send data and receive response	[strData, data content]

Python client completes interpretation and assembly of various commands sent by use cases and handles various responses of PythonServer. When designing client, take thin client as the principle, and attach importance to process (rather than processing) during use of client.

PythonServer completes interpretation and assembly of various commands sent by the client and handles various responses from Python client. When designing Server side, abstraction and induction of basic functions shall be enhanced, so as to ensure that codes can be kept at certain scale and will not be extended when functions of auto-dial system are expanded continuously.

In order to ensure smooth execution of control process between the client and the server and clear consistency in user usage, it is agreed as follows:

(1) UDP communication is adopted between the client and the server and all contents of communications are written by plaintext character strings, in order to facilitate usage and testing;

(2) A request/response mode is adopted in communication mechanism; the time required for response of requests shall not be longer than human reaction time and the response shall be accompanied with results for the commands whose results can be obtained immediately. For example, for PS service requests under normal circumstances, the commands whose response results cannot be obtained immediately can be divided into different processes, in order to ensure real-time performance of response.

3.3. Design of Python Interface of Auto-dial Testing System

Network data acquisition interface, terminal information acquisition interface, service testing and terminal control interface are Python scripting interfaces. Design of key interfaces is described in Table **1**.

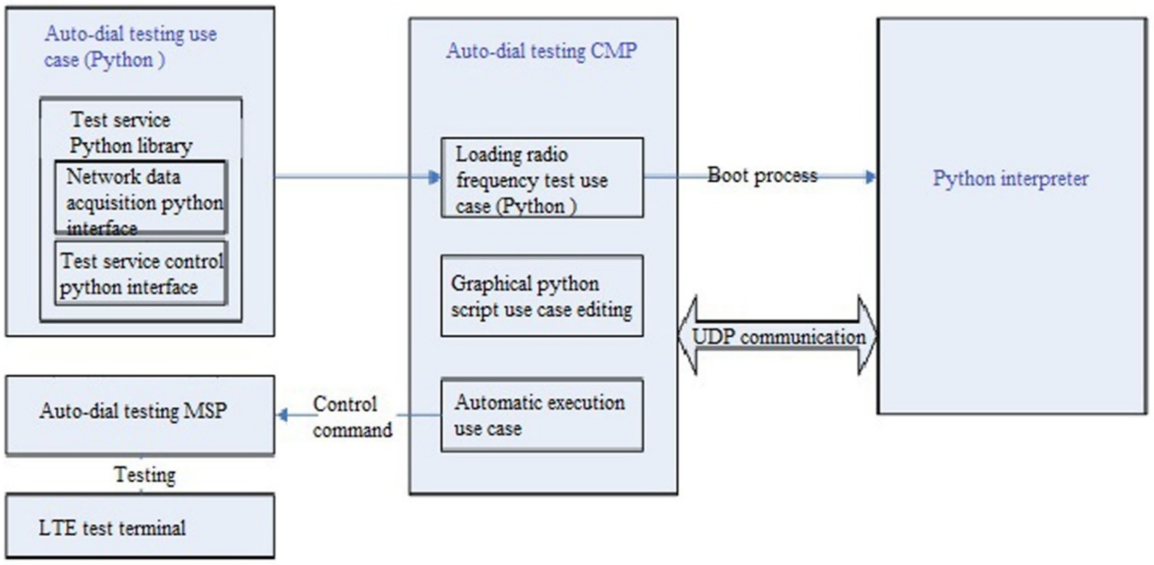
Auto-dial testing use case (Python)
Test service Python library
Network data acquisition python interface
Test service control python interface
Auto-dial testing CMP
Loading radio frequency test use case (Python)
Graphical python script use case editing
Automatic execution use case
Boot process
Python interpreter
UDP communication
Auto-dial testing MSP
Control command
Testing
LTE test terminal

Fig. (4). Python script processing flow.

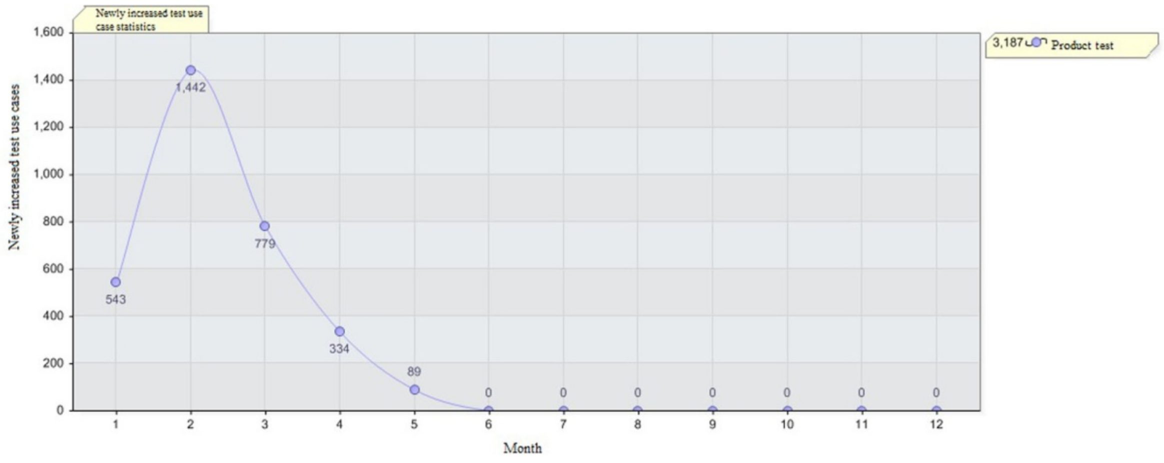

Fig. (5). Trend of newly increased use cases.

3.4. Python Script Processing Flow

Testers write the testing scripts with Python script editor or graphic interface script editor provided by CMP [4]. After writing of Python scripts is completed, CMP software loads the use cases and activates the executing processes to analyze Python scripts, as shown in Fig. (**4**).

Control module, communication module, service module, Python module and test results generation module would be involved in execution of typical testing tasks.

Taking the typical download service as an example, control module activates Python editor to execute scripts and the script command message is sent to communication module in the form of UDP after Python scripts have been downloaded to MSP. Being in real-time monitoring state, the communication module will notify Python interface module to analyze the received UDP messages. Then, the Python interface module figures out the type of service to be executed and sends it to the control module. The control module will activate the thread and calls the service module to be tested. The service module executes specific services and records desired testing results in a real-time way. After the service execution is completed, return to the control module and exit the thread. In this way, a new thread will be activated to execute corresponding testing tasks every time different messages are received. After Python interface module has resolved scripting finalization message, control module closes testing results file and be ready to execute test report uploading process.

4. SYSTEM PERFORMANCE TEST

After the Python-based LTE automatic testing system is applied, the number of newly increased testing use cases in testing of wireless network decreases from the original 1,442 to 89 (as shown in Fig. **5**) and the time spent in testing a use

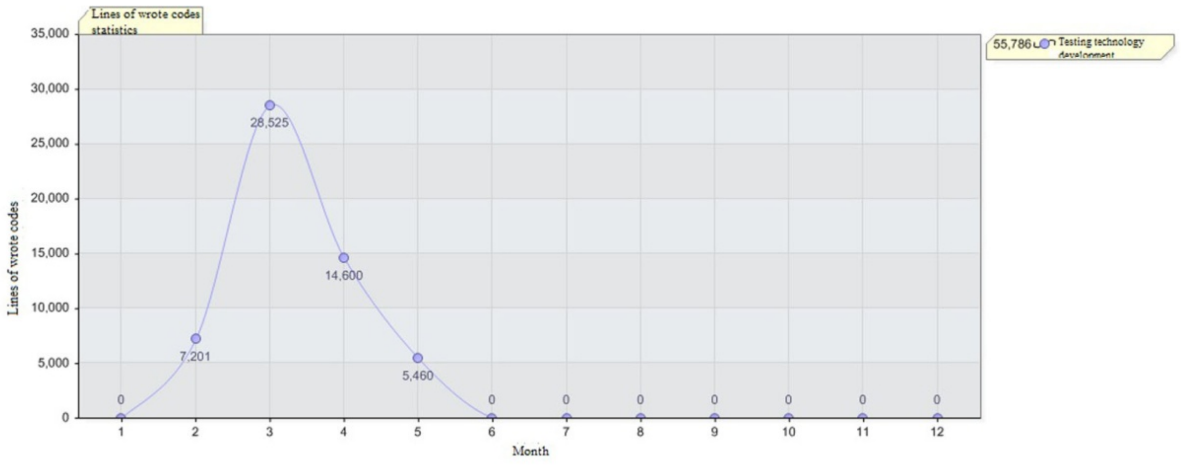

Fig. (6). Trend of lines of script code.

case suite on a regression ergodicity basis decreases from 30 days (manual test) to 12 hours now. Moreover, the number of lines of script codes written by the testers decreases from the original highest 28,525 to 5,460 as shown in Fig. (**6**).

CONCLUSION

Wireless network testing is complex and characterized by multiple scenarios and scattered indicators. As a scripting language for automatic testing, Python is simple and flexible and testers are enabled to expand their own test cases arbitrarily. Only a few simple functions need to be called without paying attention to the specific service flow. The wireless network automatic testing system as described in this paper is based on the scripts written with Python language and thus improves the testing efficiency greatly.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] l. Ferrigno, A. Pietrosanto, and V. Paciello, A Bluetooth- based proposal of instrument wireless interface", *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 1, pp. 163-170, Dec. 2005.

[2] H. Shim, I. Joe, and J. Lee, "SMS spam filtering system using SVM", *RNIS*, vol. 7, pp. 5-6, 2011.

[3] J. Luo, and M. H. Yang, "An e-mail authentication and disposable addressing scheme for filtering spam", *Journal of Cases on Information Technology*, vol. 6, no. 2, pp. 161-171, 2011.

[4] Y. Wang, and J. Cai, "Research and implementation of API test-based distributed test framework", *Computer Engineering and Design*, vol. 68, pp. 1299-1301, Jan. 2008.

Received: December 15, 2014 Revised: January 04, 2015 Accepted: February 25, 2015