The Design and Realization of One Automatic Test Framework for TVOS

Liangfu Zhao1,a, Yachao An1,b, Rui Fu1,c, Hongguo Qiu2,d, Kepeng Zheng3,e

1Academy of Broadcasting Science, SAPPRFT, The State Administration of Press, Publication, Radio, Film and Television, Beijing, China
2Device Chipset Key Technologies Platform Development Dept., HiSilicon Technologies Co., Ltd., Shenzhen, China
3Shaanxi Broadcasting Corporation, No. 336 South Road, Changan, Xi’an Shaanxi, China
azhaoliangfu@abs.ac.cn, banyachao@abs.ac.cn, cfurui@abs.ac.cn,
dqiuhongguo@hisilicon.com, e8576023@qq.com

Abstract. With the development of software industry, the framework for system software becomes more and more complex, and moves in the directions towards hierarchy, modularization and flexibility. The ensure of function and performance of the whole system through traditional manual test is low efficient and high cost. The software testing is in the process of transition from manual testing to automatic testing. This study designed and realized one automatic testing frame for TVOS which could fulfill the demand of many manufacturers and cooperative research and development. Moreover, we analyzed the testing model and designed the test cases accordingly based on the characteristics of framework of TVOS.

1. Introduction

With the booming of software industry, the software testing industry is gradually rising. Software testing is attracting more and more attentions from software developments and project managers recently, and software testing technology has become one of the hotspots in testing research. Given that the scales of software are more and more large, the degree of complexity are higher, and testing interfaces are more and more, the software testing is in the process of transition from manual testing to automatic testing. The automatic testing could make up the deficiency in manual testing, reduce the work load of manual testing, and enlarge the testing scope of software testing, which will all improve the efficiency of software testing. However, designing one automatic test framework which can be used in all software is not realistic considering that different development languages might be used for different software, and the frameworks of different software might be vastly different. Thus, the frameworks and test cases should be designed based on the specific needs of software development.

TVOS is the specific operating system used in digital television terminals in China. TVOS, which is based on Linux kernel, could manage the hardware, software and data on digital television terminals, control the activities of programs, optimize the human-computer interaction interface, and provide support for other software. Furthermore, this operating system is jointly researched and developed by cooperative teams, with different manufacturers participating in different software from bottom level
driver, middleware to application layers. In addition, the source codes and test cases were provided by different manufacturers, thus the automatic framework need to provide support for acquisition of test cases, pushing notifications and testing reports to relevant parties remotely and achieve unattended operation.

This study firstly introduced the testing model of TVOS, then designed and realized one automatic test framework for TVOS which could fulfill the demand of many manufacturers and cooperative research and development. In the end, we designed the test cases accordingly based on the characteristics of framework of TVOS.

2. The testing model of TVOS

The test framework of TVOS was hierarchical and modular. From the hierarchical point of view, TVOS is composed of the kernel layer, hardware abstraction layer, functional component layer, executive environment layer and application frame layer in a loosely coupled manner. From the modular point of view, all the five functional software layers above the kernel layer were composed of several software modules in a loosely coupled manner. Given that so many layers, modules and interfaces were included in TVOS, the comprehensive testing should be conducted from four dimensions of function, performance, security and interface to guarantee the compliance of technology and standard. The test model design could be seen in Figure 1 based on the above four testing dimensions.

![Test model of TVOS](image)

**Fig. 1 Test model of TVOS**

Function test is composed of TV live broadcast test, internet TV broadcast test, local media broadcast test, application support test and so on. Performance test is composed of boot time test, channel switching time test and so on. Security test is composed of secure boot test, application security test and so on. Interface test is composed of application frame layer test of TVOS, interface of TVOS-J and TVOS-H, component layer interface test and hardware abstraction layer interface test.

3. Design and realization of one automatic framework for TVOS

Based on the analysis of test model of TVOS, the design of automatic framework for TVOS should be fit for the situation that the development groups are distributed all over the world. The local automatic framework pays special attention to and realizes the interface test dimension in the test model. So the automatic framework should provide services like source codes synchronize, versions automatic interpretation, the regular starts of testing task, the automatic implement of testing task, the automatic generating and sending of testing report, the deployment of test cases remotely, the monitoring of...
testing environment remotely and data fusion of existing Android CTS testing frame and so on. In addition, the source codes of TVOS do not need any code for automatic test, and the automatic framework does not disrupt the integrity of devices under test to ensure the objectivity of testing results. The automatic test framework for TVOS could be seen in Figure 2.

![Automatic test framework for TVOS](image_url)

**Remote environment**  **Local unattended environment**

**Fig. 2** Automatic test framework for TVOS

The automatic test framework includes deployment of testing tasks, the automatic interpretation of software, the implement of test, sending of test reports and remote monitoring of test environment and so on.

### 3.1. The deployment of testing tasks

If test tasks are too big that one test environment center is not enough, then several test environment centers could be deployed to manage several or several kinds of devices to share or execute several test tasks side-by-side at the same time.

The deployment of testing task is the allocation and design of test map for each task environment center. Test map lists all the test cases at a certain granularity. If the deployment of test tasks is based on the result of listing of degree of granularity of module level, the test maps are operational. The test environment will download the test cases according to the test maps before the initiate of test tasks in test environment center.

### 3.2. Automatic interpretation of software

The automatic interpretation of software will be developed by scripts. The interpretation will synchronize the source codes at regular time intervals and provide the specific target codes for TVOS for each tested device accordingly.

### 3.3. The execution of test

The execution of test can be initiated by test environment center at regular intervals or immediately initiated manually. The test environment center will download test map and test cases accordingly before the start of test task from internet test case deployment server. Then the test cases will be pushed to tested devices. The push of test cases from test environment center is serial that one test case is pushed after previous test case is finished. The test environment center could also push test cases to several devices or several kinds of devices at the same time. Given that the allowances of hardware in different devices are different, that the push of test cases to different devices at the same time could test
the response of different test devices to the same test case. The test environment center will collect test results when all the test cases are finished and the complete test reports are generated and sent. The process of test implementation could be seen in Figure 3.

![Diagram](image)

**Fig. 3** Automatic test execution process

3.4. *The generation and upload of test reports*

When one test case is finished, the test result will be stored at specified path, and once the test result is accessed by test environment center, another test case will be pushed to tested device automatically. After all the test cases specified by test map are finished, the test environment center will generate a test report and upload the test report to the report server.

3.5. *The remote monitoring of test environment*

The test environment center will generate some process information during the process of test, which can be accessed by remote monitoring devices. The information could be analyzed and organized, and then presented in the form of UI by remote monitoring devices. Moreover, one remote monitoring device could monitor several test environment centers at the same time.

4. *The design and realization of test cases*

The automatic test could only be successfully realized through the cooperation between automatic test framework and test cases. Test cases include preconditions, input parameters, execution steps and expectations. Four tips should be considered during the design of test cases:

1. The effectiveness of test cases. The data input and output expectations should be clear. If the requirement is changing after several versions of software, then test cases, test data and the expected results should all change accordingly.

2. The comprehensiveness of test cases. The execution steps should be described clearly, and the order of test cases should be arranged comprehensively to improve the test execution efficiency and avoid of double test or missed test.

3. The clarity of test cases. The focus of test should be made clear and test cases should be designed accordingly for the focus of test, which include input parameters, preconditions, and expectations.
(4) The maintainability of test cases. Test cases could be maintained effectively in real time if function and requirements are changing.

Test cases of TVOS are designed according to the different needs of function layer and framework layer. Interface test cases in TVOS include JAVA application programming interface test cases, WEB application programming interface test cases, functional component layer interface test cases and hardware abstraction interface test cases.

4.1. TVOS JAVA application programming interface test cases
The TVOS Java application programming interface provide Java applications with invocation interfaces in Java object mode, and assist applications in implementing DTV services such as EPG, channel list, and TV program playing. The TVOS Java application programming interface units include unidirectional broadcast network access unit, broadcast protocol processing unit, bidirectional broadband access unit and media processing unit, and so on.

4.2. WEB application programming interface test cases
The TVOS Web application programming interface provide web applications with invocation interfaces in JS object mode, and assist applications in implementing DTV services such as EPG, channel list, and TV program playing. The TVOS Web application programming interface units include unidirectional broadcast network access unit, broadcast protocol processing unit and bidirectional broadband access unit, and so on.

4.3. Functional component layer API interface test cases
The TVOS functional component layer API interface units consist of the server and client. The TVOS functional component layer API interface units include digital TV component, media engine component and human-computer interaction component.

4.4. The TVOS hardware abstraction layer API interface test cases
The TVOS hardware abstraction layer API interface provide the upper-layer software with interfaces used to invoke the corresponding hardware capabilities. The TVOS hardware abstraction layer API interface units mainly include components of Aout, Vout, Av, Frontend, Dmx, and System.

Many test possibilities should be considered in interface test cases designing, such as normal test, abnormal test, performance test, stress test, Long and Stability test and so on (Table 1).

| Test type             | Test cases                                      | Example                          |
|----------------------|-------------------------------------------------|---------------------------------|
| Normal Test          | TestCase_ModuleName_ApplicationName_aNorm        |                                 |
| Abnormal Test        | TestCase_ModuleName_ApplicationName_aAbnm        | Input of invalid channel        |
| Performance Test     | TestCase_ModuleName_ApplicationName_aPerf        | Channel switching performance   |
| Stress Test          | TestCase_ModuleName_ApplicationName_aPress       | Repeated enter/exit of broadcasting |
| Long and Stability Test | TestCase_ModuleName_ApplicationName_aLong     | Long-time of broadcasting       |

Each test case has its own function which is independent to other test cases. All the resource taken, and hardware state which has been changed during test will be released or restored after all the test cases are finished. All the modules and components are guaranteed to meet the requirements of function and framework of TVOS based on the design and develop of test cases.
5. Conclusion
This study designed and realized one automatic test framework for TVOS, which could distribute test tasks remotely, synchronize code compilation and initiate test tasks at certain time, carry out test tasks and generate test reports automatically, and monitor the test environment remotely. The realization of test framework for TVOS could be seen in Figure 4.

![Fig. 4 Automatic test](image)

During the process of executing, this automatic framework integrates the existent internet technologies like TCP/IP protocol, FTP protocol, Samba protocol, and software tools like curl, adb, WinRAR, FileZilla. And all the internet technologies, software tools and hardware devices used in this automatic framework could have different choices. Moreover, we analyzed the test model, and designed test cases scheme accordingly based on the framework of TVOS.

References

[1] GY/T303.1-2016 Smart TV operating system - part 1: Function and architecture.
[2] Sheng Zhifan, the Smart TVOS2.0 and Broadcasting Terminal Intelligence [J], Broadcasting and TV Technology, 2016,03:34-39.
[3] Jiang Lingxian. Design and Implementation of the Web Interface Testing Automation Framework Based on Testng. [D]. Hangzhou: Zhejiang University of Technology, 2016.
[4] Liu Chunmei, Han Rui and Guo Zhichuan, the Realization of Automated Testing for the Smart TVOS Modular Layer [J], Network New Media Technology, 2016,05:45-50.
[5] Sheng Zhifan, Smart TVOS2.0—the Main Engine of Broadcasting Terminal Intelligence [J], Audio Visual Circle(Broadcasting Technology), 2016,06:47-58.