Tai-hoon Kim  Hojjat Adeli
Haeng-kon Kim  Heau-jo Kang
Kyung Jung Kim  Akingbehin Kiumi
Byeong-Ho Kang (Eds.)

Communications in Computer and Information Science 257

Software Engineering, Business Continuity, and Education

International Conferences ASEA, DRBC and EL 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, in Conjunction with GDC 2011, Jeju Island, Korea, December 2011, Proceedings

Springer
The Fractal Prediction Model of Software Reliability Based on Wavelet .................................................. 265 Yong Cao, Youjie Zhao, and Huan Wang

Source Code Metrics and Maintainability: A Case Study ................................................................. 272 Péter Hegedűs, Tibor Bakota, László Illés, Gergely Laddányi, Rudolf Ferenc, and Tibor Gyimóthy

Systematic Verification of Operational Flight Program through Reverse Engineering ............................................ 285 Dong-Ah Lee, Jong-Hoon Lee, Junbeom Yoo, and Doo-Hyun Kim

A Study on UML Model Convergence Using Model Transformation Technique for Heterogeneous Smartphone Application ........................................... 292 Woo Yeol Kim, Hyun Seung Son, and Robert Young Chul Kim

A Validation Process for Real Time Transactions ................................................. 298 Kyu Won Kim, Woo Yeol Kim, Hyun Seung Son, and Robert Young Chul Kim

A Test Management System for Operational Validation ................................................. 305 Myoung Wan Kim, Woo Yeol Kim, Hyun Seung Son, and Robert Young Chul Kim

Mobile Application Compatibility Test System Design for Android Fragmentation ........................................... 314 Hyung Kil Ham and Young Bom Park

Efficient Image Identifier Composition for Image Database ................................................. 321 Je-Ho Park and Young Bom Park

A Note on Two-Stage Software Testing by Two Teams ................................................. 330 Mitsuhiro Kimura and Takaji Fujiwara

Cumulative Damage Models with Replacement Last ................................................. 338 Xufeng Zhao, Keiko Nakayama, and Syouji Nakamura

Periodic and Random Inspection Policies for Computer Systems ............................................. 346 Mingchih Chen, Cunhua Qian, and Toshio Nakagawa

Software Reliability Growth Modeling with Change-Point and Its Goodness-of-Fit Comparisons ................................................. 354 Shinji Inoue and Shigeru Yamada

Replacement Policies with Interval of Dual System for System Transition ............................................. 362 Satoshi Mizutani and Toshio Nakagawa

Probabilistic Analysis of a System with Illegal Access ................................................. 370 Mitsuhiro Imaizumi and Mitsutaka Kimura
A Test Management System for Operational Validation

Myoung Wan Kim¹, Woo Yeol Kim²,
Hyun Seung Son², and Robert Young Chul Kim²

¹ Institute of Technology, Infinis, Inc.,
Seoul, 135-080, Korea
kimmw@infinis.com

² Dept. of CIC(Computer and Information Communication), Hongik University,
Jochiwon, 339-701, Korea
(john, son, bob)@selab.hongik.ac.kr

Abstract. NMS (Network Management System) is a central monitoring system that can manage equipment on network environments. This should be used for efficient and centralized management of network equipment. On NMS, it enables the real-time transmission and monitoring of the data on states, problems, composition, and statistics of equipment that make a network. But we need to verify whether it works on operations or functions of NMS operations or not. To do this, this paper suggests a test management system for the efficient verification of NMS environments. In order to develop a test management system, requirements from each NMS shall be extracted, and designed and materialized based on them. The suggested system enables efficient test management, result analysis, and comparative verification of test versions.

Keywords: NMS, test, test management system, verification.

1 Introduction

NMS (Network Management System) is a central monitoring system that can manage equipment on a network. This is used for efficient and centralized management of network equipment [1,2]. NMS enables to the real-time transmission and monitoring of the data on states, problems, composition, and statistics of equipment that make a network. When problems with the equipment are occurred, an alarm signal can be transmitted to a manager for a speedy measurement. Statistics and states of networks can also be analyzed based on the collected information [3].

Figure 1 shows the network of NSM. Each company manages its own network, which should make NMS system test system requirements, functions, and operations.

In order to verify NSM of various environments, the existing test has been conducted manually via a checklist with the managed results based on documentation [4,5,6]. There are the problems with manual operation of the test results in that it is difficult to integrate test results as various testers conduct testing simultaneously because the analysis and management of the test results does not appropriately executed. The version management of test cases is also not sufficient and the reuse of test cases is difficult.
This paper suggests test management system to solve such problems. In order to develop a test management system, each NMS environment is analyzed, which helps to extract requirements. Based on the extracted requirements, a test management system is designed and managed. The suggested system enables efficient test management, result analysis, and comparative verification of test versions. The currently used NMS tests were conducted as a case study and the results were applied to the suggested test management system.

This paper consists of the following. Chapter 2 explains about NMS as related studies. Chapter 3 describes the suggested test management system. Chapter 4 shows the application system of this suggested test management system as an applied case study. Finally, Chapter 5 provides conclusions and describes the future studies.

2 Network Management System

NMS centrally monitors communication networks on the network [7]. NMS consists of NMS servers providing main functions, which collect receiving data from equipment, sending them to NMS Server, and saving data to DE server. NMS is organized as Figure 2. Each UTM equipment on the network transmits system monitoring data, alarm data related to problems, log data, and equipment registration data to NMS system via collector server while NMS Server provides central management of the related network.
NMS Server can centrally manage network equipment based on the information on UTM equipment states, problems, and log received from Collector Server. It provides with user management UI for managing Collector Server and UTM equipment. It analyzes the data received from UTM equipment and provides statistical information. Based on the collected and analyzed data, it organizes the status of interface links of UTM equipment and IPSEC tunnel links as well as the monitoring screen, generating a report.

Individual connection to each UTM equipment can bring the information of setup of interface, routing, NAT, firewall, IPS, web filter, and content filter and the setup change is available. Multiple UTM equipments can be grouped to order firewall rules, group IPS rules, group web filter rules, and content filter rules simultaneously.

3 Proposed Test Management System

3.1 The Test Management System Architecture

Figure 3 is architecture of the suggested test management system. The values to be fed firstly in the test management system include that the version information, types of test manuals, and test cases are extracted through analysis. The user authorization is implemented via the account management module. After authorization, the information is managed in the TESTID management mode.

After the test cases are extracted through analyses, the relevant test cases are tested while the results are managed through the test manual management module. The test manual management module can manage preconditions, testing steps, and actual results. Finally, the testing for test results is managed in the test result management module, which provides the functions such as the result document printing and preview.
3.2 Integrated data model

The integrated data model is to manage the data generated in the test management system. Figure 4 is an expression of this integrated data model in the E-R diagram.

Fig. 4. Integrated data model for a test management system
The integrated data model integrates the basic TESTID information, list of preconditions, list of testing stages, and list of actual results, enabling the printing of test result forms through the test manual management module and the test deliverables management module in Figure 5.

**Fig. 5. Generation of the test results**

The specifications on the basic TESTID information, list of preconditions, list of testing stages, and list of actual results were defined while the Pseudo Code of data-integrating algorithm in Figure 6 was prepared based on the defined classification codes. The data-integrating algorithm can lead to the extraction of test results.

```java
String report = A1 + A2 + A5 + A3 + A4
If( B1.length > 0 ){ 
    for( B1.length ){ 
        report += B1
    }
}

If( C1.length > 0 ){ 
    for( C1.length ){ 
        report += C2 + C1
        If( D1.length > 0 ){ 
            for( D1.length ){ 
                report += D3 + D2 + D1
            }
        }
    }
}
report += A7
If( E1.length > 0 ){ 
    for( E1.length ){ 
        report += E2 + E1 + E3
    }
}
```

**Fig. 6. Data-integrating algorithm**
4  NMS Verification Using the Test Management System

This chapter explores the functions of the developed test management system and verifies it by using the test management system and carrying out NSM tests based on the actual test cases.

The followings take place to verify the test management system.

1) COMM Manager testing
2) Feeding the test results in the test management system
3) Confirming the printing of test results in the test management system

4.1 COMM Manager Testing

In order to bring the firewall rule setup list from NMS Server, the firewall rule list is transmitted based on the communication between COMM Manager in Collector Server and COMM Agent in UTM equipments. Afterwards, the test confirmation in NMS Server user’s UI will be carried out.

1) Testing a request for the list of firewall rules to COMM Agent of the relevant UTM equipment from NMS Server to COMM Manager

```
[COMM Manager] FilterCmd called!!!

[COMM Manager] new create FPolicyInfo called!!!
```

2) Before importing the firewall rules from COMM Agent, testing an authorization request from COMM Manager to the relevant UTM equipment for security authorization

```
[COMM Manager] GET /auth/loginUser_id=---&pwd=--- HTTP/1.1
Host: 10.88.1.216:9251
User-Agent: WEB/2.0
Keep-Alive: 100
Connection: keep-alive
```

3) ID and PW to be checked in the UTM equipment and testing the transmission of authorization messages to COMM Manager

```
13:40:37.959589 IP 10.0.0.1 15.2557 > 10.90.1.216 loc: S 601706224 601706241(0) win 66666 len:
S: 1460 scr: 1460 mtu: 1487 len: 1460
P: 10.0.0.1 15.2557
```

4) Testing an authorization message receipt from COMM Agent of the UTM equipment in COMM Manager
5) Testing a request for firewall rules from COMM Manager to the UTM equipment’s COMM Agent

6) Testing the transmission of the firewall list from the UTM equipment’s COMM Agent to COMM Manager

7) After receiving the firewall rules from COMM Manager to the UTM equipment COMM Agent, testing the transmission to NMS Server

8) Testing the confirmation of the relevant UTM equipment’s firewall rule list in NMS Server user’s UI
4.2 Confirmation of the Printing of Test Results

The results of tests of each stage are fed into the test management system. Pressing Preview or Print ensures printing based on the following form in Figure 7 through the test result management module.

<table>
<thead>
<tr>
<th>TEST manual ID</th>
<th>Functional test-FW-List of rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Listing firewalls</td>
</tr>
<tr>
<td>TSR</td>
<td>NET-SYS-FWList</td>
</tr>
<tr>
<td>Test environment</td>
<td>Network environment</td>
</tr>
<tr>
<td>Precondition</td>
<td>Precondition-SYS-login</td>
</tr>
</tbody>
</table>

Test stages

1) A request for a list of firewalls to the relevant UTM equipment's COMM Agent from NMS Server to COMM Manager.

2) Before bringing firewalls from COMM Agent, COMM Manager request for the authorization to the relevant UTM equipment for security authorization.

3) ID and FW shall be checked from the UTM equipment and the authorization message is sent to COMM Manager.

4) Receive the authorization message from the UTM equipment's COMM Agent from COMM Manager.

5) COMM Manager request for the list of firewalls to the UTM equipment's COMM Agent.

6) The list of firewalls is transmitted to COMM Manager from the UTM equipment's COMM Agent.

7) COMM Manager receives the list of firewalls from the UTM equipment's COMM Agent and sends it to COMM Manager.

8) Check the list of firewalls of the relevant UTM equipment in NMS Server User UI.

Fig. 7. Test result form
5 Conclusion

The test to verify the efficient operation of NMS was manually conducted based on a documented checklist. The test results were managed in Excel or Word, requiring the analysis or management of test results. Many testers tested each assigned part and collected all the results, resulting in problems with integrated management. This produced requirements for the test case version management and reuse.

This paper designed and materialized the test management system to overcome such problems with NMS. The integration became more convenient as many testers’ test results were centrally managed via the test management system. As the test results were documented and kept after computation, the safe storage was ensured. The test management system enabled separate saving of test results by version while the test case reuse are helped with efficient testing. Also, the interrelatedness of test cases was found while the automated report printing for test results became possible.

While there is a system that manages erroneous operation (bugs, errors, or faults) by using a tool like Bugzilla in the existing open sources and Bugzilla tool can be conveniently used as a means for communication between developers and testers, the suggested test management system not only provides overall test management of the system but also organically derives the result documents according to circumstances.

The test management tool and the test automation tool can be integrated to derive a way to support more efficient testing as a future study.

Acknowledgments. This research was supported by 2010 Hongik University Research Fund and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation.

References