*Journal of Software Engineering and Applications*, **2011, 4, 688-699**
doi:10.4236/jsea.2011.412081 Published Online December 2011 (http://www.SciRP.org/journal/jsea)

# Automated Testing Framework for ODBC Driver

**Dan Ye**

State Grid Electric Power Institute, China Real-Time Database Co. Ltd., Nanjing, China.
Email: yedan2@sgepri.sgcc.com.cn

Received October 14th, 2011; revised November 23rd, 2011; accepted December 5th, 2011.

## ABSTRACT

*At first, the core ideology, advantage and principle of Software Testing Automation Framework (STAF) are presented in this paper. Several automated testing frameworks are summarized and analyzed. In addition, date driven automation test framework is given more attention. Test script is the important composing part of software test automation. Then this paper introduces several technologies of script along with their characteristics. Every technology applies to different places. Moreover, an automated test system with an automated test script language XML which is implemented to solve the problems in current automatic testing solutions. The design and implantation of the test script automation framework based on apache ant is put forward. The proposed key technology details of framework are also introduced. We take the automation test methodology of database connectivity operations by ODBC data source as example for validation, using auto test script as the key technology. Finally, the results demonstrate that it can best increase software test effectiveness and reduce workloads as well as save more efforts.*

***Keywords*: *Software Automation Testing, Universal Test Script, XML, Data-Driven, Apache Ant***

## 1. Introduction

With testing system become more and more complicate, software function is enriched according to this requirement. Based on object-oriented framework, it is the driven for innovative software engineering technology and modern automation test tools in the development of automation test system. The software reuse and development efficiency is the hotspot in the research field. For the sake of improving the utility of test script, the universal design is the research hotspot of automation test field around the world for reducing the cost of automation testing system. Software is the core of automation testing system, the task of develop the universal test script is very important [1]. Software testing automation is an efficient way of software quality and testing effectiveness. Framework is the reuse design of the whole and part system which is the interactive method [2]. A great framework can achieve code reuse and top level design reuse while we can finish the program by extent and cut the framework. Based on the framework development program can reduce the workload and improve development efficiency.

In this paper, the automation testing framework based on apache ant is proposed. The framework of verification on SQL parse model can achieve high levels of reuse. The rest of paper is organized as follows: Section 2 introduces test framework of automation. Section 3 discusses test script and test script model. Section 4 proposes software design and system architecture implementation. Section 5 draws the conclusion.

## 2. The Test Framework of Automation

### 2.1. Definition

The definition of automated test framework is the set of practice of hypothesis and conceptual. The software testing technology can improve the efficiency. The automated test framework can raise the overall efficiency of testing and reduce the software drawbacks on function and performance [3].

### 2.2. Core Thought

The core thought of the automation is as follows [4]:

1) The separation of application program and automation test script.

2) The separation of test description and detail realization.

3) The separation of test script and test data.

### 2.3. Traits

Efficient automation test script should satisfy the following traits [5]:

1) The separation of test framework and application program.

2) The test framework easy to extend and maintain.

 

3) The language of test script is independent.

4) The test script is easy to use and have low computational complexity.

### 2.4. The Principle of Automation

During the testing period, repeated work will need automation test while test with GUI will need manual test. Test procedure is as follows:

1) Test design. The design of test cases and requirement, feature design and testability, case design, test strategy, test arrangement, test environment are composed of test design.

2) Result comparisons. Analysis the comparison of test script if it is passed.

3) Test reports generate. By making collection of test results generate test report, we can obtain the analysis of failed test result by tester and developer.

### 2.5. Basic Software Test Automation Framework

The following five test frameworks will be demonstrated. There are test modularity framework, test library framework, keyword driven, table driven test framework and data driven test framework, as well as mixed framework.

The test modularity framework needs creating small and independent model as well as test script of application program. The tree structure of little test script is composed of test script. As far as the automated test framework, keyword-driven test can change into table driven test. These datasets and keyword automation is independent with test tool. In keyword driven testing, function of application program and each test execution steps are put together. The test framework produces a batch of test cases by few codes, which is reused by using datasets. Data driven test is a framework [6]. Input and output data is abstract from data file. In this framework, variable can put input and verification output. During the whole program, test script can read data file and save test status and information, by capture tool or manual code script load into the variable. In table driven test, test case is included in data file. In data driven test, data file includes test data.

Data driven test automated framework which is Plug and play architecture include system independent, product independent, data independent, order independent. During test system, test data and test order change, there is no need to change source code, data driven design support the reuse of model. Data driven test automation methodology is useful for common duplicated test cases. It can reduce the workload of coding. The test framework is created test scenario and test data by xml documentation to make test case automation based on the realization of all functions. Data driven by parse data file which is needed by test cases.

The test automation framework is TAF model including framework and common model, professional model. Framework includes some common functions. Test combination is the set of test case which is aimed to accomplish test object.

## 3. Test Script and Test Script Model

Based on test system and requirement analysis, propose an extended and reusable automation test system which can improve software reusability and develop efficiency. The following part will base on the analysis of script.

- Linear script: take notes of manual execution test script. Structure linear script and control script execution instructions. These instructions are controlled structure.
- Sharing script: one or more scripts are used by multiple test cases. In every test cases, call these scripts can fulfill the task. The sharing of script is the key criterion of automation test case [7].

There are following merits:

1) Save the time and make script high efficiency.

2) If duplicated tasks make changes, it is needed to change a script.

3) Script modular structure design.

Using linear and sharing script, make test automation while data driven test will add many test cases. Data driven test is suitable for large scale testing. Keyword driven script is to change data file into test cases, by using a series of keywords [8]. The keyword exists in test file. Controlled script read the keyword in test file. Using keyword call related supporting script. Therefore, testing method is need to change test script instead of test case when test tool changes.

The keyword driven script is a kind of structure script, controlled script and supporting script is to translate keyword. All the information includes into dataset. The highlight of this technology is the separation of test information and realization.

The traits of script technology are well commentaries and rational construction, easy understanding, documenttation and well maintained [9]. The easier scripts programming, the more drawbacks during the automation test.

### 3.1. Based on XML Test Script

The generation steps of XML test script are focused on the definition of XML model, obtain the prepared program using XML test script generator to make XML test script. XML is the high level expression of test cases which has more advantages [10-17]:

1) Script on the form of XML with easy modify and handle with.

2) Test code can build and execute to speed up the testing efficiency on a large scale.

3) Generation test analysis and log documentation besides from test code.

 

4) When abstract processing and interface structure information, the definition of extended XML model make XML test script more universal without redesign on specific model.

### 3.2. Example of Test Code

To make the test analysis of requirements specification is the first step [18]. Then setup parameter is another basic unit during the test processing. Further, make automation test software universal, which can be divided into several layers [19].

1) Script generation model

2) Script running model

In the following, an example of test implementation of ODBC data source for database connectivity is demonstrated [20]. Test script can be run on the operation system of windows 7 and windows server 2008. Test code is based on C++.

Code list 1.1 is part of test code for batch insert 20 W into real-time database by ODBC data source [21-23].

**Code list 1.1 ODBC_BatchInsertPoint_20W.cpp**

```
// ODBC_BatchInsertPoint_20W.cpp -- Batch Insert 20W Point
sret=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&henv);
if(!isSuc(sret))    printf("apply environment handle error \n");
sret=SQLSetEnvAttr(henv,SQL_ATTR_ODBC_VERSION,(void*)SQL_OV_ODBC3,0);
sret = SQLAllocHandle(SQL_HANDLE_DBC,henv,&hdbc);
if(!isSuc(sret))    printf("apply for connection handle error \n");
printf("hdbc address before entry:%x\n", &hdbc);
fstream fsr(result, fstream::in | fstream::out | fstream::trunc);
sret=SQLConnect(hdbc,(SQLCHAR*)serverName,strlen(serverName),(SQLCHAR*)user,strlenuser),(SQLCHAR*)password,
strlen(password));
cout << "SQLConnect return: " << sret << endl;
sret = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hsmt);
cout << "results of applying for statement handle：" << sret << endl;
        if(hsmt == NULL)
        {
                printf("error hstmt\n");
        }
sret = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hsmt);
        cout << "Statement handle return：" << sret << endl;
        if(hsmt == NULL)
        {
                printf("error hstmt\n");
        }
        int BatchNum=200000;
        NAME *name = new NAME[BatchNum];
        unsigned char *archive=new unsigned char[BatchNum];
        Deadband *deadband=new Deadband[BatchNum];
SQLUSMLINT*ParamStatusArray=newSQLUSMALLINT[BatchNum];
SQLINTEGER *nameLenArray = new SQLINTEGER[BatchNum];
SQLINTEGER *archiveLenArray = new SQLINTEGER[BatchNum];
SQLINTEGER *deadbandArray = new SQLINTEGER[BatchNum];
for (int i = 0; i < BatchNum; i++)
        {
                sprintf(name[i], "'HS0100000%d'", i);
                archive[i] = 1;
                nameLenArray[i] = SQL_NTS;
                archiveLenArray[i] = 0;
                deadband[i] = 2;
        }
Clock_t start = clock();
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_BIND_TYPE, SQL_PARAM_BIND_BY_COLUMN, 0);
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAMSET_SIZE, &BatchNum, 0);
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_STATUS_PTR, ParamStatusArray, 0);
sret=SQLBindParameter(hsmt,1,SQL_PARAM_INPUT,SQL_C_CHAR, SQL_CHAR, 32, 0, (SQLPOINTER)name, 32, nameLenArray);
sret=SQLBindParameter(hsmt,2,SQL_PARAM_INPUT, SQL_C_BIT, SQL_BIT, 1, 0, (SQLPOINTER)archive, 0, archiveLenArray);
sret=SQLBindParameter(hsmt,3,SQL_PARAM_INPUT,SQL_C_FLOAT, SQL_FLOAT, 1, 0, (SQLPOINTER)deadband, 0, deadbandArray);
printf("addpoint 20W...\n");
sret = SQLPrepare(hsmt, (SQLCHAR*)userinputsql, SQL_NTS);
sret = SQLExecute(hsmt);
        Clock_t end = clock();
        double elapse = (end - start);
        printf("elapse: %f\n", elapse);
         fsr << sret << endl;
         fsr.close();
        if (SQL_SUCCESS != sret)
        {
                char SqlState[6];
                int NativeError = 0;
        char ErrorMessage[SQL_MAX_MESSAGE_LENGTH];
                int EMLength = SQL_MAX_MESSAGE_LENGTH;
                int RealLength = 0;
SQLGetDiagRec(SQL_HANDLE_STMT,hsmt,1,(SQLCHAR*)SqlState,(SQLINTEGER*)&NativeError,(SQLCHAR*)ErrorMessage,EMLength,(SQLSMALLINT*)&RealLength);
                printf("sqlstate:%s\n", SqlState);
                printf("error message:%s\n", ErrorMessage);
        }
        SQLFreeHandle(SQL_HANDLE_STMT, hsmt);
        SQLDisconnect(hdbc);
        SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
        SQLFreeHandle(SQL_HANDLE_ENV, henv);
        return 0;
        printf("batch insert 20W successful！ ");
```

Code list 1.2 is part of test code for batch insert 100 W into real-time database at single point.

**Code list 1.2 ODBC_BatchInsertPointValue_100W.cpp**

```
// ODBC_BatchInsertPointValue_100W.cpp -- Batch Insert 100W PointValue
sret=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&
```

 

```
henv);
sret=SQLSetEnvAttr(henv,SQL_ATTR_ODBC_VERSION,(void*)SQL
_OV_ODBC3,0);
sret = SQLAllocHandle(SQL_HANDLE_DBC,henv,&hdbc);
sret=SQLConnect(hdbc,(SQLCHAR*)serverName,strlen(serverName),
(SQLCHAR*)user,strlenuser),(SQLCHAR*)password,strlen(password));
sret = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hsmt);
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_BIND_TYPE,
SQL_PARAM_BIND_BY_COLUMN, 0);
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAMSET_SIZE,&BatchNu
m, 0);
sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_STATUS_PTR,
ParamStatusArray, 0);
sret=SQLBindParameter(hsmt,1,SQL_PARAM_INPUT,SQL_C_CHAR,
SQL_CHAR, 32, 0, (SQLPOINTER)name, 32, nameLenArray);
        int BatchNum = 1000000;
        int k=6;
       unsigned long *ids=new unsigned long[ BatchNum ];
        TIME * time=new TIME[ BatchNum ];
       unsigned long *status = new unsigned long[ BatchNum ];
       float *value = new float[ BatchNum ];
SQLUSMALLINT *ParamStatusArray = new SQLUSMAL-
LINT[ BatchNum ];
       SQLINTEGER *idsArray = new SQLINTEGER[ BatchNum ];
       SQLINTEGER *timeArray = new SQLINTEGER[ BatchNum ];
       SQLINTEGER *statusArray = new SQLINTEGER[ BatchNum ];
       SQLINTEGER *valueArray = new SQLINTEGER[ BatchNum ];
       for (int i = 0; i < BatchNum; i++)
       {
              ids[i] = k;
              sprintf(time[i], "'2011-8-25 19:30:01.%d'", i);
              status[i] = i;
              value[i] = -i;
       }
        clock_t start = clock();
 sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_BIND_TYPE,
SQL_PARAM_BIND_BY_COLUMN, 0);
 sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAMSET_SIZE,
&BatchNum, 0);
 sret=SQLSetStmtAttr(hsmt,SQL_ATTR_PARAM_STATUS_PTR,
ParamStatusArray, 0);
sret=SQLBindParameter(hsmt,1,SQL_PARAM_INPUT,SQL_C_ULON
G, SQL_INTEGER, 1, 0, (SQLPOINTER)ids, 0, idsArray);
 sret=SQLBindParameter(hsmt,2,SQL_PARAM_INPUT,
SQL_C_CHAR, SQL_CHAR, 31, 0, (SQLPOINTER)time, 32, timeAr-
ray);
 sret=SQLBindParameter(hsmt,3,SQL_PARAM_INPUT,
SQL_C_ULONG, SQL_INTEGER, 1, 0, (SQLPOINTER)status, 0,
statusArray);
 sret=SQLBindParameter(hsmt,4,SQL_PARAM_INPUT,
SQL_C_FLOAT, SQL_FLOAT, 1, 0, (SQLPOINTER)value, 0, val-
ueArray);
 sret = SQLPrepare(hsmt, (SQLCHAR*)userinputsql, SQL_NTS);
       printf("userinputsql:%s\n",userinputsql);
       printf("add value 100W into pointid=6...\n");
 sret = SQLExecute(hsmt);
       clock_t end = clock();
       double elapse = (end - start);
       printf("elapse: %f\n", elapse);
       fsr << sret << endl;
       fsr.close();
        for (int m = 0; m < BatchNum; m++)
       {
          fss << value[m]<< "\t" << status[m]<< "\t" << time[m] << endl;
          fss.close();
       }
       if (SQL_SUCCESS != sret)
       {
              char SqlState[6];
              int NativeError = 0;
              char ErrorMessage[SQL_MAX_MESSAGE_LENGTH];
              int EMLength = SQL_MAX_MESSAGE_LENGTH;
              int RealLength = 0;
SQLGetDiagRec(SQL_HANDLE_STMT, hsmt, 1, (SQL-
CHAR*)SqlState,(SQLINTEGER*)&NativeError,(SQLCHAR*)ErrorM
essage, EMLength, (SQLSMALLINT*)&RealLength);
              printf("sqlstate:%s\n", SqlState);
              printf("error message:%s\n", ErrorMessage);
       }
              SQLFreeHandle(SQL_HANDLE_STMT, hsmt);
              SQLDisconnect(hdbc);
              SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
              SQLFreeHandle(SQL_HANDLE_ENV, henv);
              system("pause");
              Sleep(50000);
              return 0;
}
```

## 4. Apache Ant Tool

Apache ant is a java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run java applications. Ant can also be used effectively to build non java application, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks.

The Apache Ant project is part of the Apache Software Foundation. Ant is different from other build tools. Instead of a model where it is extended with shell-based commands, Ant is extended using java classes. Instead of writing shell commands, the configuration files are XML-based, calling out target tree where various tasks get executed. Each task is run by an object that implements a particular task interface.

### 4.1. The Basic Usage of Apache Ant

Apache ant's build files are written in XML. Each build file contains one project and at least one target. Targets contain task elements. Each task element of the build file can have an id attribute and can later be referred to by the value supplied to this. The value has to be unique.

A project has three attributes:

 

[Attribute]

Name: the name of the project.

Default: the default target to use when no target is supplied.

Basedir: the base directory from which all path calculations are done. This attribute might be overridden by setting the "basedir" property beforehand. When this is done, it must be omitted in the project tag. If neither the attribute nor the properties have been set, the parent directory of the build file will be used.

Each project defines one or more targets. A target is a set of tasks you want to be executed. When starting Ant, you can select which target you want to have executed. When no target is given, the project's default is used.

[Targets]

A target can depend on other target. You might have a target for compiling, for example, a target for creating a distributable. You can only build a distributable when you have complied first, so the distribute depends on the compile target. Ant resolves these dependencies. It should be noted, however, the Ant's depends attribute only specifies the order in which targets should be executed, it does not affect whether the target that specifies the dependency get executed if the dependent target did not run.

### 4.2. Script Implementation Architecture

A task is a piece of code that can be executed. A task has multiple attributes. The value of an attribute might contain references will be resolved before the task is executed. All tasks share a task name attribute. The value of this attribute will be used in the logging messages generated by Ant.

The Automation script of whole structure is presented in **Figure 1**. **Figures 2-8** indicates separately files structure and subfolder structure.

**Figure 2** depicts that the overall structure of file composition. The configuration file and xml file are included into the cases folder. The file composition structure of the folder of Cases, which includes the API, log, tools as well as the list of configuration file and xml execute document. In the first level folder, build.xml need manual configuration for test requirement. Each function suite is written in xml document in the build.xml file. Casesuit.ini is written for the definition of the abbreviation of the xml document name. Build.xml is the main running document including ODBC Batch Insert Value suite, ODBC Select Order suite, ODBC Add point suite, ODBC Update all point suite, ODBC Select mode suite and ODBC delete all point as well as XLS report, which defines the running order of suite. The Doc folder structure is shown in **Figure 3**. Meanwhile, the composition of Etc folder is illustrated in **Figure 4**. Besides, the composition of Bin folder is shown in **Figure 5**. Record result presents file which store the result and the expected result of case as well as file to record case report. The Lib folder structure is demonstrated in **Figure 6**. The API folder structure includes all the interface execution programs as shown in **Figure 7**. Moreover, the composition of Log folder is illustrated in **Figure 8**. The log folder demonstrates the result-list in each according case file where include sourefile.txt and result.ini. The sourcefile.txt includes the input parameter value and the result.ini is the judgment result of the test execution. In addition, the MD5 compare tool and record result tool as well as the result generation tool in the tools folder. Md5 checksum of the source file and checksum of the destination file as well as the file to log the compared result.

During the research of ODBC operations, use of XML technology construct XML model which is maintained well to improve software flexibility and extendable.

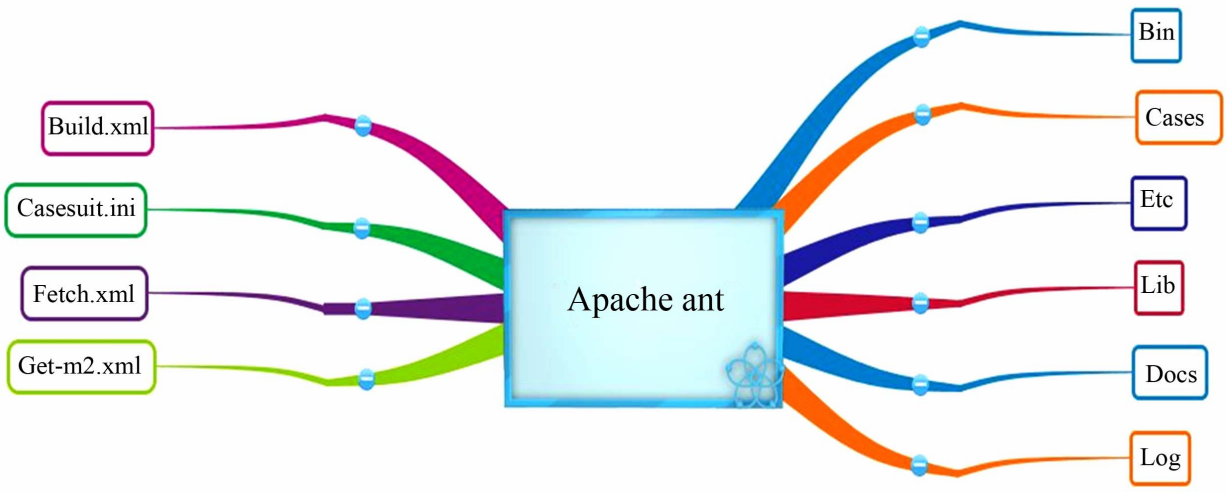


**Figure 1. Automation script structure.**

 

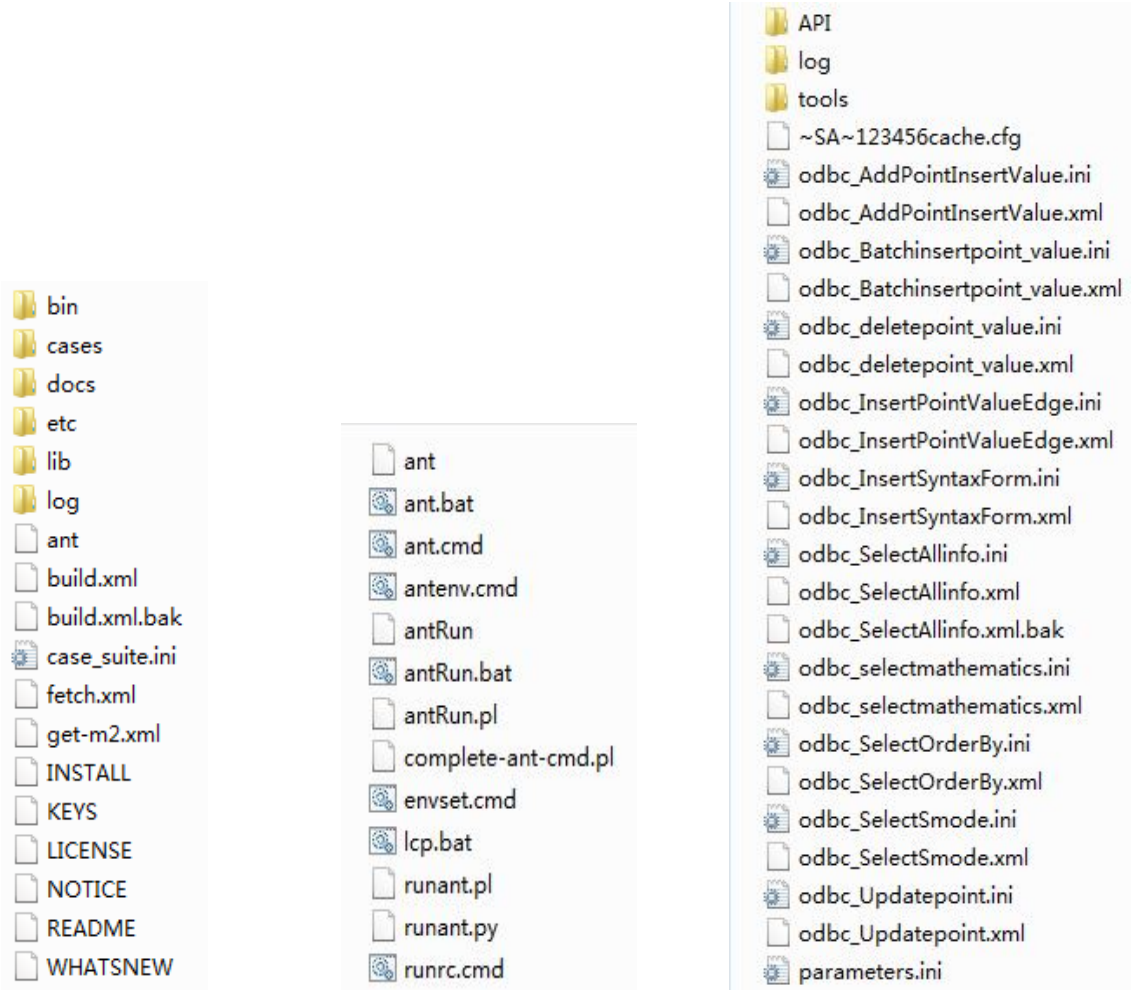


**Figure 2. The overall structure of file composition.**

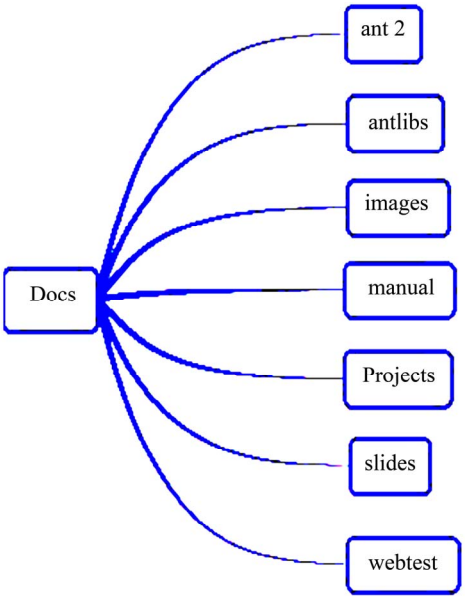


**Figure 3. The composition of Doc folder.**

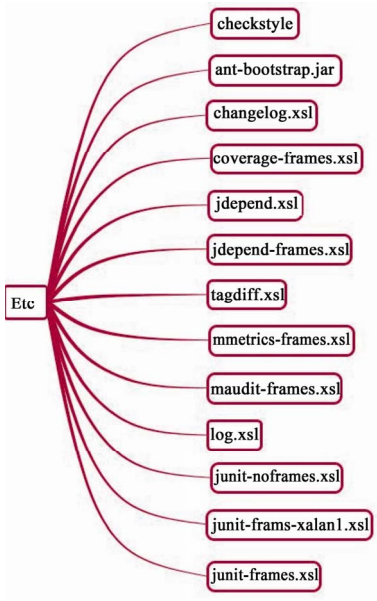


**Figure 4. The composition of Etc folder.**

 

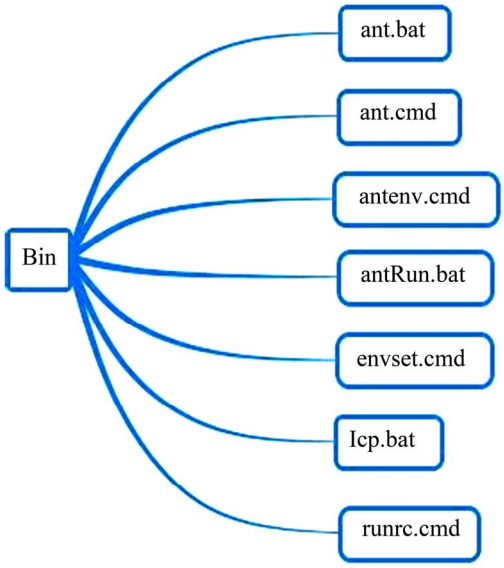


**Figure 5. The composition of Bin folder.**

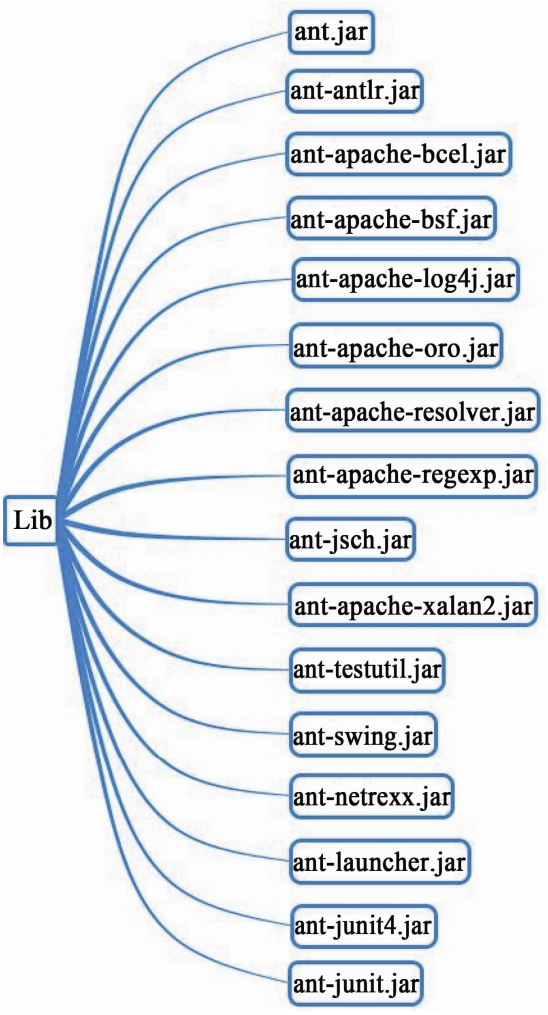


**Figure 6. The composition of Lib folder.**

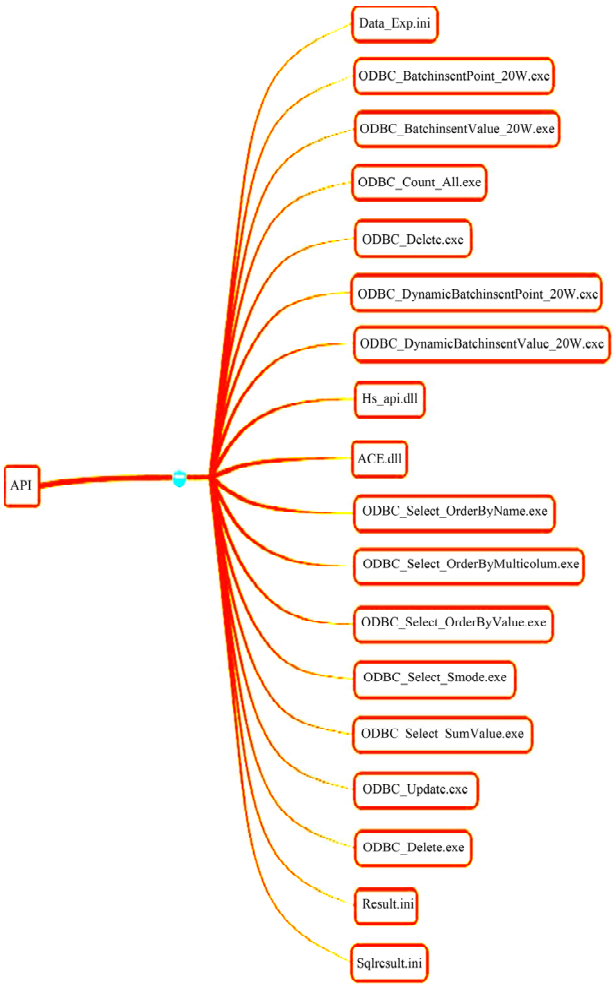


**Figure 7. The composition of structure.**


**Figure 8. The composition of Log folder.**

## 4.3. Test Script Implementation

Test program generation tool is based on XML script, C++ language code is tested. Test is composed of test cases manage, test plan, test script writing, test execution and test result. Tester makes test script according to the steps of test case.

By using the test script generation model, test requirement can be changed into test script which can be taken into automation test tool. The following part is XML segment in the configuration file.

Code list 1.3 **[build.xml]**

 

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ODBCAutomation" basedir="." default="build">
<property name="bin.dir" value="bin"/>
<property name="log.dir" value="log"/>
<property name="case.dir" value="cases"/>
<property name="temp.log.dir" value="cases\log"/>
<property name="tool.dir" value="cases\tools"/>
<property file="case_suite.ini"/>
<property file="${case.dir}\parameters.ini"/>
<target name="logFile">
        <tstamp>
<format property="touch.time"pattern="MM-dd-yyyy hh:mmaa"
offset="0" unit="hour"/>
        </tstamp>
    </target>
<target name="odbc_Batchinsertpoint_value"depends="logFile">
            <sequential>
<echo message="****** Start to run ODBC Batch Insert Value
suite ******"/>
   <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
<argvalue="${case.dir}\${odbc_bipv.xml}"/> </exec>
 <echo message="****** Finish to run ODBC Batch Insert
Value suite ******"/>
                    </sequential>
    </target>
<target name="ODBC_Selectsmode"depends="odbc_Batchinsert
point_value">
            <sequential>
<echo message="****** Start to run ODBC Select Smode suite
******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
       <arg value="${case.dir}\${odbc_ss.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Select Smode suite
******"/>
                    </sequential>
    </target>
<target
name="ODBC_SelectOrderBy"depends="ODBC_Selectsmode">
            <sequential>
<echo message="****** Start to run ODBC Select OrderBy suite
******"/>
    <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
<arg value="${case.dir}\${odbc_sob.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Select OrderBy
suite ******"/>
                    </sequential>
    </target>
<target
name="odbc_SelectAllinfo"depends="ODBC_SelectOrderBy">
            <sequential>
<echo message="****** Start to run ODBC Select All info suite
******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
 <arg value="${case.dir}\${odbc_sai.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Select All info
suite ******"/>
                    </sequential>
    </target>
<target
name="ODBC_Updatepoint"depends="odbc_SelectAllinfo">
            <sequential>
<echo message="****** Start to run ODBC Update All Point
info suite ******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
       <arg value="${case.dir}\${odbc_up.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Update All Point
info suite ******"/>
                    </sequential>
    </target>
<target
name="ODBC_Selectmathematics"depends="ODBC_Updatepoin
t">
            <sequential>
<echo message="****** Start to run ODBC Delete All Point
suite ******"/>
       <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
<arg value="${case.dir}\${odbc_smm.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Delete All Point
suite ******"/>
                    </sequential>
    </target>
<target
name="odbc_deletepoint_value"depends="ODBC_Selectmathem
atics">
            <sequential>
<echo message="****** Start to run ODBC Delete All Point
suite ******"/>
       <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
       <arg value="${case.dir}\${odbc_dpv.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Delete All Point
suite ******"/>
                    </sequential>
    </target>
<target
name="odbc_AddPointInsertValue"depends="odbc_deletepoint_v
```

 

```
alue">
            <sequential>
<echo message="****** Start to run ODBC Add Point suite ******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
         <arg value="${case.dir}\${odbc_apiv.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Add All Point suite ******"/>
                    </sequential>
    </target>
<target name="odbc_InsertPointValueEdge" depends="odbc_AddPointInsertValue">
                     <sequential>
<echo message="****** Start to run ODBC insert Point value Edge suite ******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
       <arg value="${case.dir}\${odbc_ipve.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC insert Point value Edge suite ******"/>
                    </sequential>
    </target>
<target name="ODBC_DeleteAllPoint"depends="odbc_InsertPointValue Edge">
            <sequential>
<echo message="****** Start to run ODBC Delete All Point suite ******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
                        <arg value="ant"/>
                        <arg value="-f"/>
<arg value="${case.dir}\${odbc_dpv.xml}"/>
                       </exec>
<echo message="****** Finish to run ODBC Delete All Point suite ******"/>
                    </sequential>
    </target>
<target name="Move_Logs"depends="ODBC_DeleteAllPoint">
    <tstamp>
<formatproperty="log.time"pattern="yyyyMMdd-HH-mm-ss"
                        offset="0" unit="hour"/>
          </tstamp>
        <mkdir dir="${log.dir}\${log.time}"/>
        <move todir="${log.dir}\${log.time}">
             <fileset dir="${temp.log.dir}"></fileset>
        </move>
        <sleep seconds="5" />
    </target>
 <target name="Report_Gen" depends="Move_Logs">
              <sequential>
<echo message="****** Start to generate final XLS report ******"/>
      <exec executable="cmd" failonerror="true">
                        <arg value="/c"/>
          <arg value="${tool.dir}\Report_Gen"/>
      <arg value="${basedir}\${log.dir}\${log.time}"/>
                        </exec>
<echo message="****** Finish to generate final XLS report ******"/>
                    </sequential>
    </target>
    <target name="build" depends="Report_Gen">
<echo message="Run all automation suites successfully!"/>
    </target>
</project>
```

Which <target> shows the definite test object and <depends> defines the test object which depends on that test object in the above code.

### 4.4. Configuration File

Some test parameters need change often according to test requirement. These parameters save in a file. While test script running, test file can be read by tested parameter information. When running different test cases, it is enough to modify the configuration file. In the example, setup defined parameter in the file <casesuit.ini>.

Code list 1.4 **[casesuit.ini]**

```
[Deploy_Case_Suite_Xml]

odbc_apiv.xml=odbc_AddPointInsertValue.xml

odbc_bipv.xml=odbc_batchinsertpoint_value.xml

odbc_dpv.xml=odbc_deletepoint_value.xml

odbc_ipve.xml=odbc_InsertPointValueEdge.xml

odbc_isf.xml=odbc_InsertSyntaxForm.xml

odbc_sai.xml=odbc_SelectAllinfo.xml

odbc_sob.xml=odbc_SelectOrderBy.xml

odbc_ss.xml=odbc_selectsmode.xml

odbc_smm.xml=odbc_selectmathematics.xml

odbc_up.xml=odbc_updatepoint.xml
```

Code list 1.5 **[odbc_Batchinsertpoint_value.xml]**

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ODBCAutomation" basedir="." default="deploy">
<property name="api.dir" value="API"/>
<property name="log.dir" value="log"/>
<property name="tool.dir" value="tools"/>
<property file="parameters.ini"/>
<property file="odbc_Batchinsertpoint_value.ini"/>
<target name="logFile">
```

 

```
<tstamp>
<formatprop-
erty="touch.time"pattern="MM-dd-yyyyhh:mmaa"offset="0"
  unit="hour"/>
</tstamp>
</target>
<target name="ODBC_BatchInsertValue100W_case1"depends=
"logFile">
<!-- run batchinsertpoint case -->
<sequential>
<echo message="****** Start to run odbc Batch Insert value
100W test case 1 ******"/>
<mkdir dir="${log.dir}\1"/>
<exec executable="cmd" failonerror="true">
<arg value="/c" />
<argvalue="${api.dir}\ODBC_BatchInsertPoint_20W"/>
<arg value="${1.serverName}" />
<arg value="${1.user}" />
<arg value="${1.password}" />
<arg value="${1.sql}" />
<arg value="${basedir}\${log.dir}\${1.result}"/>
</exec>
<exec executable="cmd" failonerror="false">
<arg value="/c"/>
<arg value="${api.dir}\ODBC_BatchInsertValue_100W"/>
<arg value="${2.serverName}" />
<arg value="${2.user}" />
<arg value="${2.password}" />
<arg value="${2.sql}" />
<arg value="${basedir}\${log.dir}\${2.result}"/>
<arg value="${basedir}\${log.dir}\${1.InsertValue_Src}"/>
</exec>
<exec executable="cmd" failon error="true">
<arg value="/c"/>
<arg value="${tool.dir}\Record_Result"/>
<arg value="${basedir}\${log.dir}\${1.result}"/>
<arg value="${basedir}\${log.dir}\${2.result}"/>
<arg value="${1.expect.result}"/>
<arg value="${log.dir}\${odbc_batch insertpoint_value.report}"/>
</exec>
<echo message="****** Finished to run odbc Batch Insert value
100W test case 1 ******"/>
<sleep seconds="10"/>
</sequential>
</target>
<target name="ODBC_BatchInsertValue20W_case2"depends=
"ODBC_BatchInsertValue100W_case1">
<!-- run connection case -->
<sequential>
<echo message="****** Start to run ODBC Batch Section Insert
value 20w test case 2 ******"/>
<mkdir dir="${log.dir}\1"/>
<exec executable="cmd" failonerror="true">
<arg value="/c" />
<arg value="${api.dir}\ODBC_BatchInsertPoint_20W"/>
<arg value="${1.serverName}" />
<arg value="${1.user}" />
<arg value="${1.password}" />
<arg value="${1.sql}" />
<arg value="${basedir}\${log.dir}\${1.result}"/>
</exec>
<exec executable="cmd" failonerror="false">
<arg value="/c"/>
<arg value="${api.dir}\ODBC_BatchInsertValue_20W"/>
<arg value="${3.serverName}" />
<arg value="${3.user}" />
<arg value="${3.password}" />
<arg value="${3.sql}" />
<arg value="${basedir}\${log.dir}\${3.result}"/>
<arg value="${basedir}\${log.dir}\${2.InsertValue_Src}"/>
</exec>
<echo message="****** Finished to run ODBC Batch Section
Insert value 20w test case 2******"/>
<sleep seconds="10"/>
</sequential>
```

Code list 1.6 **[odbc_Batchinsertpoint_value.ini]**

```
[Test ID 1]
1.serverName=127.0.0.1:9000:HS-1
1.user=SA
1.password=123456
2.serverName=127.0.0.1:9000:HS-1
2.user=SA
2.password=123456
1.sql="Insert into pointinfo(name,archive,deadband) values(?,?,?);"
2.sql="Insert into pointvalue values(?,?,?,?);"
1.result= 1\\result.ini
2.result= 2\\result.ini
1.InsertValue_Src=1\\sourefile.txt
1.expect.result = 0
[Test ID 2]
1.serverName=127.0.0.1:9000:HS-1
1.user=SA
1.password=123456
3.serverName=127.0.0.1:9000:HS-1
3.user=SA
3.password=123456
1.sql="Insert into pointinfo(name,archive,deadband) values(?,?,?);"
3.sql="Insert into pointvalue values(?,?,?,?);"
1.result= 1\\result.ini
3.result= 3\\result.ini
2.InsertValue_Src=2\\sourefile.txt
2.expect.result = 0
```

Code list 1.7 **[parameters.ini]**

```
[System]
serverName="127.0.0.1:9000:HS-1"
user="SA"
password="123456"
[Report]
odbc_batchinsertpoint_value.report=odbc_batchinsertpoint_value.txt
odbc_deletepoint_value.report=odbc_deletepoint_value.txt
odbc_updatepoint.report=odbc_updatepoint.txt
odbc_selectsmode.report=odbc_selectsmode.txt
odbc_selectmathematics.report=odbc_selectmathematics.txt
odbc_dynamicinsert.report=odbc_dynamicinsert.txt
odbc_selectorderby.report=odbc_selectorderby.txt
odbc_selectallinfo.report=odbc_selectallinfo.txt
```

In the code list 1.4, casesuit.ini depicts the abbreviation of substitution file name. As can be seen in code list 1.5, odbc_batchinsertpoint_value.xml is for batch insert value 100W, where it includes the parameters.ini and odbc_batchinsertpoint.ini. Besides, server name, user and password, SQL items are written in this configuration file. First target is the task for batch insert 100 W value at one point. Second target is the task is for batch insert 20 W

 

value at each section. As for code list 1.6, odbc_batchinsertpoint_value.ini includes the details of configuration information. Code list 1.7 describes the configuration file parameters.ini which includes the abbreviation of report file name and the relevant configuration of database user information.

### 4.5. Test Experimental Analysis

As for every test case, test output has its own test case analysis. Check if it is satisfy with standard output to define the result is passed whether or not.

When running the command line to execute build. xml by the usage of ant, the test result will be shown automatically step by step. All the test result will be screened in the execute window. During the test execution, each task execution result indicates that all the database operation successful.

By the analysis for test result, it can verify that it can best increase software test effectiveness and reduce workload as well as save more efforts.

## 5. Conclusions and Future Work

In this paper, the automation test system based on automation script generation is proposed. By the separation of test data and script, make test case template. Let test script reuse. Through the automation test script, automation script can generate quickly and accurately. With more and more test cases, there are more and more test codes, bugs can be modified in the test procedure. Test builds use automation test show its advantage. The test tool can run on multi-automation test platform. Meanwhile, the automation test tool can finish the regression test of automation. Besides, the automation test tool can be applied in the unit test and integrate test. In the future work: (1) strengthen the usage of automation test tool (2) support Linux, Solaris platform using the test tool.

## REFERENCES

[1] D. J. Mosley and B. A. Posey, "Software Test Automation. Just Enough Software Test Automation," B. Zheng L. J. Huang and Q. C. Cao Translated, Machinery Industry Press, Beijing, 2003.

[2] W. Frakes and K. Kang, "Software Reuse Research: Status and Future," *IEEE Transactions on Software Engineering*, Vol. 31. No. 7, 2005, pp. 529-536. doi:10.1109/TSE.2005.85

[3] W. Aresi and F. McGarry, "Defining Leverage Points for Increasing Reuse," *Minnowbrook Work Shop on Software Reuse*, Blue Mountain Lake, New York, 1987.

[4] M. R. Qureshi and S. A. Hussain, "A Reusable Software Component-Based Development Process Model," *Advances in Software Engineering*, Vol. 39, No. 2, 2008, pp. 88-94. doi:10.10161/j.advengsoft.2007.01.021

[5] C. J. M. Geisterfer and S. Ghosh, "Software Component Specification: A Study in Perspective of Component Selection and Reuse," *Fifth International Conference on Commercial-off-the-Shelf* (*COTS*) *Based Software Systems*, Washington, 13-16 February 2006, p. 9.

[6] M. Kelly, "Choosing a Test Automation Framework," 2003. http://www.128.ibm.com/developerworks/rational/library/591/html#N10223

[7] N. Carl, "Test Automation Frameworks," 2000. http://safsdev.sourceforge.net/FRAMESDataDrivenTestAutomation Frameworks.htm

[8] H. H. Sthamer, "The Automatic Generation of Software Test Data Automatically," *Proceedings of the International Conference on Automated Software Engineering*, Linz, 20-25 September 2004, pp. 346-349.

[9] J. Al-Dallal and P. Sorenson, "Testing Software Assets of Framework based Project Families," *During Application Engineering Stage Journal of Software*, Vol. 3, No. 5, 2008, pp. 11-25.

[10] S. A. Yahia, N. Koudas, A. Marian, D. Srivastava and D. Toman, "Structure and Content Scoring for XML," *Proceedings of the* 31*st VLDB Conference*, Trondheim, August 2005, pp. 362-372.

[11] T. S. Kin, J. H. Lee, J. W. Song and D. H. Kim, "Similarity Measurement of XML Documents Based on Structure and Contents," *International Conference on Computational Science* (*ICCS*), Springer-Verlag, Heidelberg, 2007, pp. 902-905.

[12] S. C. Haw and C. S. Lee, "Twing INLAB: A Decomposition-Matching—Mering Approach to Improving XML Query Processing," *American Journal of Applied Sciences*, Vol. 5, No. 9, 2008, pp. 1199-1205. doi:10.3844/ajassp.2008.1199.1205

[13] V. Mihajlovic, G. Ramirez, T. Wester Veld, D. Hiemstra, H. E. Blok and A. P. de Vries, "Vague Element Selection, Image Search, Overlap, and Relevance Feedback," *Lecture Notes in Computer Science*, Vol. 3977, 2006, pp. 72-78. doi:10.1007/11766278_6

[14] P. O. Gilivie and J. Callan, "Parameter Estimation for a Sample Hierarchical Generative Model for XML Component Retrieval," *Lecture Notes in Computer Science*, Vol. 3977, 2006, pp. 211-224.

[15] B. Sigurbjournsson, J. Kamps and M. de Rijke, "The Effect of Structured Queries and Selective Indexing on XML Retrieval," *Lecture Notes in Computer Science*, Vol. 3977, 2006, pp. 104-118.

[16] B. Jeong, D. Lee, H. Cho and J. Lee, "A Novel Method for Measuring Semantic Similarity for XML Schema Matching," *Expert Systems with Applications*, Vol. 24, 2008, pp. 1651-1658. doi:10.1016/j.eswa.2007.01.025

[17] H. Z. Wang, J. Z. Li, W. Wang and X. M. Lin, "Coding-Based Join Algorithm for Structure Queries on Graph-Structured XML Document," *World Wide Web*, Vol. 11, No. 4, 2008, pp. 485-510. doi:10.1007/s11280-008-0050-4

[18] D. Ranjan and A. K. Tripathi, "Variablity-Based Models for Testablity Analysis of Frameworks," *Journal of Soft-*

 

*ware Engineering and Applications*, Vol. 3, No. 6, 2010, pp. 455-459. doi:10.4236/jsea.2010.35051

[19] D. D. Black, M. E. C. Hull and K. Jackson, "Systems Engineering and Safety—A Framework," *Software*, Vol. 5, No. 1, 2011, pp. 43-53.

[20] G. Butler, "Object-Oriented Frameworks," 15*th European Conference on Object-Oriented Programming*, Tutorial Budapest, 18-22 June 2001, pp. 1-70.

[21] J. Al-Dallal and D. Sorenson, "Estimating the Coverage of the Framework Application Reusable Cluster-Based Test Cases," *Information and Software Technology*, Vol. 50, No. 6, 2008, pp. 595-604. doi:10.1016/j.infsof.2007.07.006

[22] T. Connolly and C. Begg, "Database Systems: A Practical Approach to Design, Implementation and Management," Pearson Education, Upper Saddle River, 2004.

[23] X. Wu and J. Feng, "A Framework and Implementation of Information Content Reasoning in a Database," *WSEAS Transactions on Information Science and Applications*, Vol. 6, No. 4, 2009, pp. 579-588.

 