An Automatic Testing Method for GUI Using the Framework of Three-Layer Test Script

Maosheng Huang [a, *] and Letian Zeng [b]

Software Quality Engineering Research Center, The fifth electronic research institute of MIIT, Guangzhou, 510610, China.

[a, *] huangms@ceprei.com, [b]zengletian@ceprei.com

Abstract. To reduce the maintenance workload of test script and implement a prompt multi-version regression tests, this paper proposes a novel automatic testing method for GUI based on three-layer script and gives the detailed implementation steps, among which the three-layer script includes reusable script, object mapping script and test script. This method makes it possible for the development and test to be carried out in parallel to obtain a rapid iteration for the software. Meanwhile, it can dramatically improve the reusable ration of the test script and effectively implement the reuse of test script under the condition of multi-project and multi-version, which can be regarded as an effective method of improving the test success rate and investment-return rate for automatic software testing. Experiments are presented to demonstrate the feasibility and simplicity of the proposed method.

Keywords: Automatic Software Testing; Three-Layer Test Script; Framework; Data-Driven.

1. Introduction

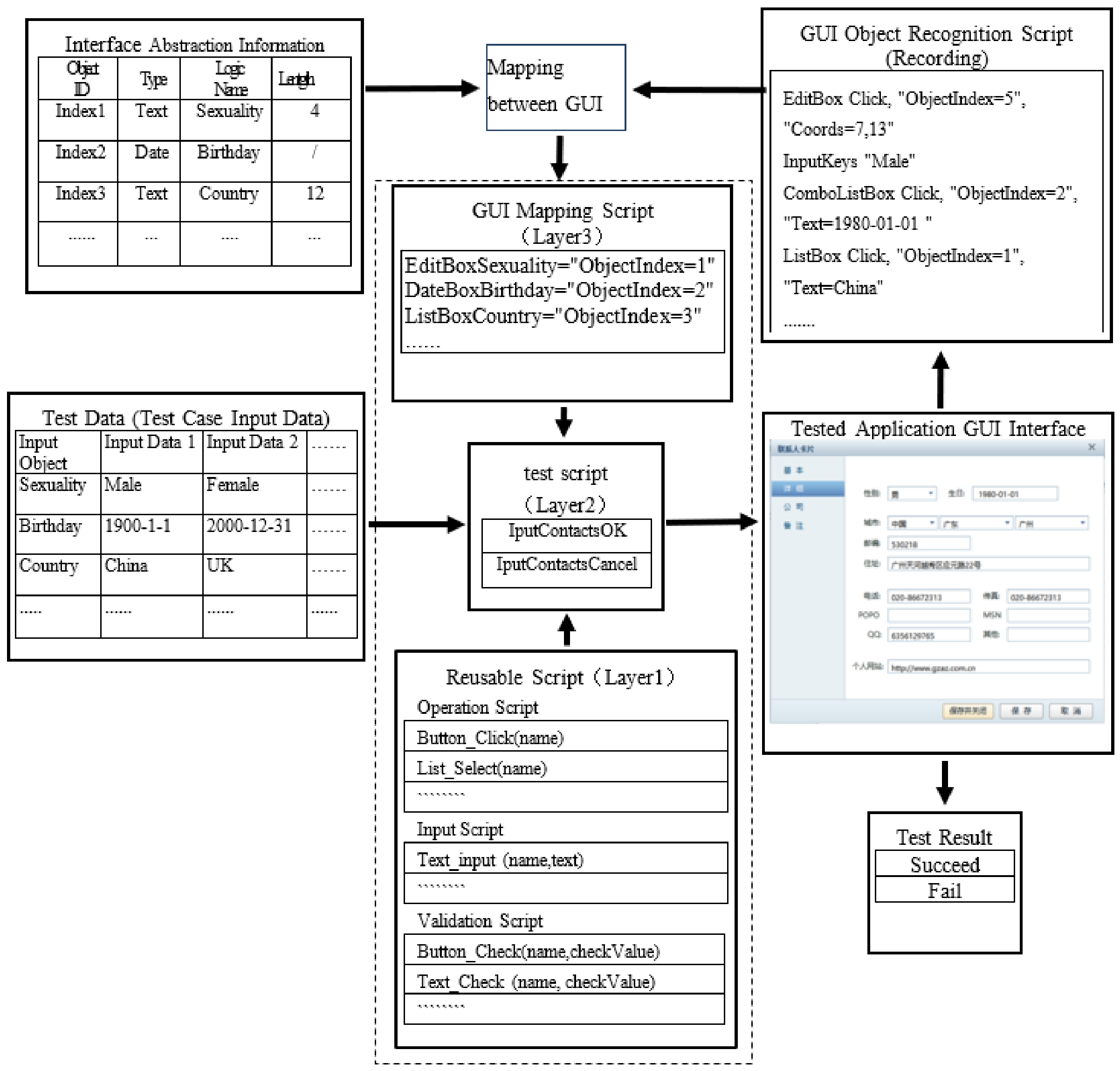
With a larger scale and more complex structure for the software, the amount of software testing work is growing rapidly. While more time is required for the software testing, the delivery cycle of the project is much shorter than before [1]. Moreover, a great deal of repeated work for the regression testing has become a necessity when it comes to the applications of agile development techniques and the prompt iterations of the software. Manual testing cannot satisfy the requirement of regression testing for graphical user interface (GUI), but automatic testing is an effective way to solve this problem.

In recent years, a lot of organizations paid attention to automatic software testing, however, 80% of the automatic testing trials has failed [2-4]. Therein, one of the main reasons for these failures is that the more modifications of the software requirements with prompt substitutions for different software versions, the more workload for repeated regression testing, which will result in a deep decrease of the return on investment ratio for automatic testing. To deal with this problem, a novel automatic testing method for GUI is proposed using the three-layer script framework, which is consisted of reusable layer (Layer1), testing script (Layer2) and GUI mapping script (Layer3), respectively.

2. Automatic Test Framework of GUI

2.1 Design of Test Script Framework

The automatic testing script framework for GUI is illustrated in Fig. 1, which is consisted of reusable script (Layer 1), test script (Layer 2) and GUI mapping script (Layer 3). The role and assignment of the scripts in each layer are depicted in Table 1.



Interface Abstraction Information

Object ID	Type	Logic Name	Length
Index1	Text	Sexuality	4
Index2	Date	Birthday	/
Index3	Text	Country	12
......	...	....	...

GUI Object Recognition Script (Recording)

```
EditBox Click, "ObjectIndex=5",
"Coords=7,13"
InputKeys "Male"
ComboListBox Click, "ObjectIndex=2",
"Text=1980-01-01 "
ListBox Click, "ObjectIndex=1",
"Text=China"
.......
```

GUI Mapping Script (Layer3)

```
EditBoxSexuality="ObjectIndex=1"
DateBoxBirthday="ObjectIndex=2"
ListBoxCountry="ObjectIndex=3"
......
```

Test Data (Test Case Input Data)

Input Object	Input Data 1	Input Data 2	......
Sexuality	Male	Female	......
Birthday	1900-1-1	2000-12-31	......
Country	China	UK	......
.....	......	......	......

test script (Layer2)

```
IputContactsOK
IputContactsCancel
```

Reusable Script (Layer1)

```
Operation Script
  Button_Click(name)
  List_Select(name)
  ````````
Input Script
  Text_input (name,text)
  ````````
Validation Script
  Button_Check(name,checkValue)
  Text_Check (name, checkValue)
  ````````
```

Figure 1. GUI automatic test framework for three-layer script

1)Reusable script (Layer1)

As an underlying script, the reusable script can be shared not only in different GUI for a single project, but also among various projects. For common operations like text recording, menu selection and list box content selection, a universal script, GUI widget ID and associated operation data (clicking, input, pitching on) can be designed as input parameters of the script for each kind of operation, respectively. Generally, the reusable script can be classified as operation script, input script and validation script. It can be specially developed the automatic testing engineers in advance and merged into a reusable script library for diverse GUI testing scripts [5,6].

2)Test script (Layer 2)

The test script is composed of several reusable scripts (Layer1) which is according to specific business processes or operation procedures. Also, it is a scripted implementation of operation procedures of the test cases and tested input data. By calling reusable script (Layer1) and GUI mapping script (Layer3), the test script can carry out automatic testing by controlling the software, inputting tested data, validating the tested execution result
```

Table 1. Role assignment of three-layer script

Script	Definition	Script Application
Reusable Script (Layer1)	As the underlying script, the reusable script can be classified as operation script, input script, validation script and so on. Also, the reusable script is mainly used to implement some common testing operation, such as text recording, menu selection, list box content selection, button click and validation for expected output.	Due to the fact that the utilization and testing for all software are the combinations of these basic operations, the reusable script is consistent among different projects. However, the only difference is that the tested data used are read from diverse data files. Therefore, these scripts can be shared in different projects, which can dramatically reduce the workload of the script maintenance via decreasing the number of the script with a sharing method.
Test Script (Layer2)	By calling reusable script and object, the test script can map the script, control software, input the test data and validate the testing result.	Aiming at specific script for tested software project, the test script carries out the testing for specific function points and business function. The main functions include the following steps: reading tested data from the test data file, calling the reusable script as well as the object mapping script, importing these data to specific object of the tested software, validating the coincidence of the tested results and the expected results, recording the Log and Bug.
GUI Mapping Script (Layer3)	The GUI mapping script implements the mapping connections between the GUI object logic name and the real GUI object of the tested software.	GUI mapping script is the key of automatic software testing and makes it possible for test engineers and automatic test script developers to work in parallel, implementing the synchronization of test development and software development. Finally, GUI is employed to map the script.

3)GUI mapping script (Layer 3)

The GUI mapping script (Layer3) is generated by the mapping between the interface abstraction information and the recorded object recognition script via GUI object and object ID. It constructs the mapping relationship of the application GUI interface object for the submitting test and the interface abstraction information from expected test cases. Also, it can provide the relevance mapping of executing test cases and inputting tested data for test script (Layer2), implementing a data-driven automatic testing.

2.2 Implementation of Test Script Framework

Software testing model is the framework for software testing work and can be employed to instruct the software testing process. It is a significant assurance for guaranteeing the quality and effectiveness of the testing [7-9]. Common testing models contain V model, W model and so on [8], in which GUI testing is the main work during the system testing stage and the acceptance testing stage [7,9], as illustrated in Fig. 2 and Fig. 3. Besides, the GUI automatic testing framework based on three-layer script can be seamlessly matched with common software testing process models, such as V model and W model.

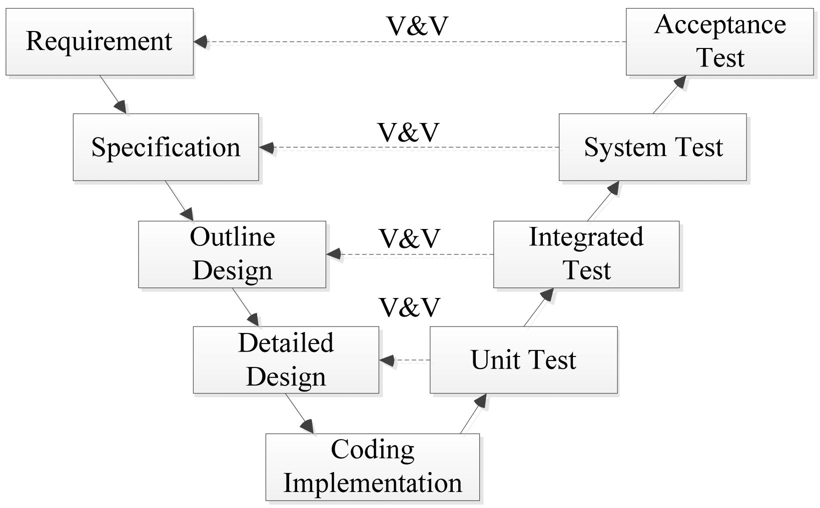


Figure 2. V model for software testing

In accordance with the design of the GUI automatic testing framework, software development, software testing, automatic testing design make an agreement on the abstraction for the GUI object of the tested object so that they can advance work in parallel during the stages of requirement and preliminary design within V model and W model. On the basis of software requirement specification and design interpretation document, developers design the software and encode and the testing engineers carry out test design, designing test cases and test data. Meanwhile, the automatic testing engineers do the designing work of test script and reusable script. Since the tested objects are submitted to execute the system testing and the regression testing, the automatic testing engineers recognize ID by recording GUI objects, construct the mapping among object IDs of the interface abstractions. So, the mapping relationship is built between the logic name of the interface object and real interface object of the tested software, as illustrated in Fig. 1. Also, the automatic testing executions and the expected results are compared. The implementation of GUI automatic testing framework is irrelevant to specific testing tools and applicable to common automatic testing tools, such as Winrunner, Robot and son on. The test process model stage and the role assignment implemented by GUI automatic testing framework are shown in Table 2.

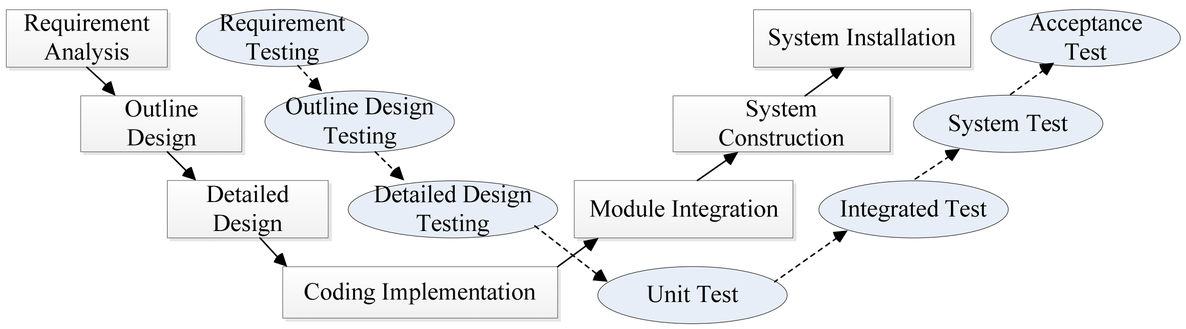


Figure 3. W model for software testing

Table 2. Role assignment comparison table for test process model stage and GUI automatic test framework

Test Process Model Stage	Role	Work Content and Achievement	Results Application	GUI Automatic Test Framework
Requirement Analysis	Requirement Analysis	Determining the input, output and their corresponding flags	Test Design	
	Test Execution	Designing test cases and determining the operation steps, input data as well as output data	Script Design	Test Data (Test Case)
	Script Design	Scripting test cases and generating test script	Test Execution	Test Script
Design and Coding	Software Development	Developing and coding		
	Test Execution	Going on designing test cases and determining steps for test operations, input data and output data	Script Design	Test Data (Test Case)
	Script Design	Going on scripting test cases and generating scripts; adding reuse script design of specified type according to GUI design	Test Execution	Test Script Reusable Script
System Test and Acceptance Test	Software Development	Submitting the tested application	Test Execution	
	Test Execution	Adding more test cases	Script Design	Test Data (Test Case)
	Script Design	Recording GUI object script, designing object mapping script, combining test script designed before with tested object	Test Execution	GUI Mapping Script

3. Implementation Process

The flowchart of GUI automatic testing framework mainly includes six stages, which are listed in the following as Fig. 4 shows: object abstraction and convention for interface input; test case design; test script design; data-driven implementation; recognition, recording and mapping for interface object; test execution and regression test.

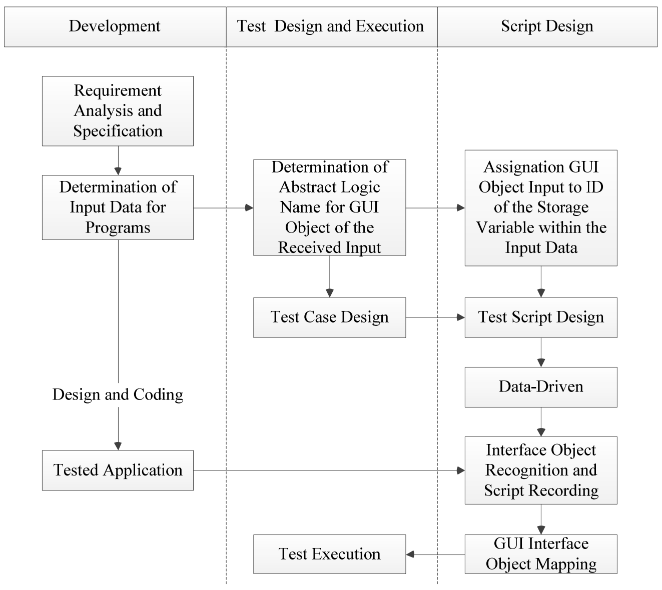

Development
Test Design and Execution
Script Design
Requirement Analysis and Specification
Determination of Input Data for Programs
Determination of Abstract Logic Name for GUI Object of the Received Input
Assignation GUI Object Input to ID of the Storage Variable within the Input Data
Test Case Design
Test Script Design
Data-Driven
Design and Coding
Tested Application
Interface Object Recognition and Script Recording
GUI Interface Object Mapping
Test Execution

Figure 4. Automatic testing flowchart for GUI

3.1 Object Abstraction and Convention for Interface Input

Since the (software requirement specification) achieved by analysing the requirement, software developer, testing engineers and automatic script design engineers come to an agreement with GUI object ID of the received input object by extracting the data content of the software program according to business and function requirement. Then, testing engineers design the test cases and the automatic script design engineers make use of the agreed logic name of the interface object to write the test script. Finally, the test cases and the test script are correlated into an organic automatic testing script set via GUI object mapping script.

3.2 Test Case Design

According to the aforementioned abstractions and the conventions, testing engineers edit the tested data, containing tested input data and expected results, with the agreed data recognition format and order. At last, operation steps and tested data make up the test cases.

3.3 Test Script Design

The script includes the reusable script and the tested script. Generally, the reusable script can be classified into operation script, input script as well as validation script. As the underlying script, the reusable script is mainly used to carry out some common test operations, such as text content recording, menu selection and list box content selection, button click and validation for expected output. It is a kind of universal script which can be applied in many projects. Therefore, the reusable script library can be called and extended to satisfy the requirements of the specific project. For the customized test script of the project, the automatic script design engineers give the test script of the test cases designed by testing engineers on the basis of the agreed logic names of the interface object as well as the storage variable ID of the input data.

3.4 Data-driven Implementation

The aim of data-driven implementation is to separate the script from the data, making it possible for a script to test several groups of the tested data, improving the flexibility and reuse degree [10, 11] of the tested script. Several test cases can share one and the same test script by the following steps: testing engineers isolating the operation steps of the test cases and the tested data; automatic script design engineers using the variables to replace the inputs of the script; recording the values of the variables from the associated data files.

3.5 Recognition, Recording and Mapping for Interface Object

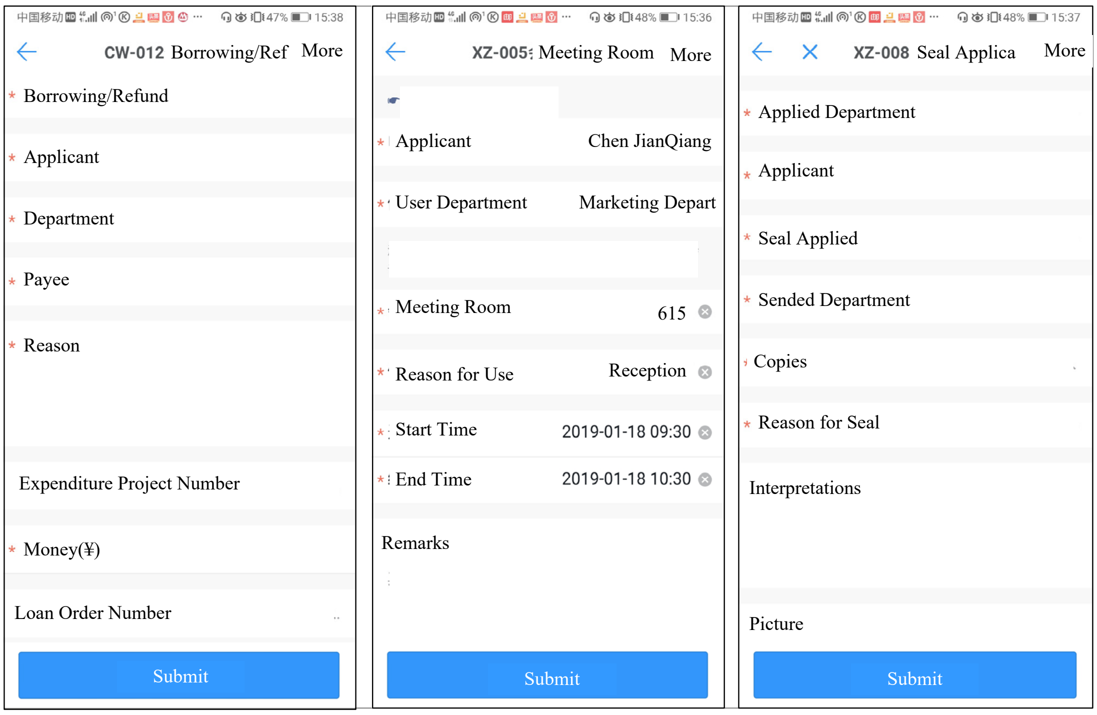
When the program is developed and submitted to be tested, the interface object is recognized via the recording function of the testing tools, extracting the recognition identifier of the program interface object in the recording script. Hence, the mapping relationship is constructed between the logic name and the recognition identifier of the interface object within the recording script. That is, the recognition identification of the interface object and the logic name of the above-determined interface object can be correlated by the testing tools. The correlated content is included in the execution script. The test cases, automatic test script, drive data and application object can be correlated into an organic integrity. At the moment, an automatic testing set comes into being.

3.6 Test Execution and Regression Testing

After constructing the test environment, the testing engineers begin to execute the automatic test cases, analyse the result data and fix the bugs. During the regression test, new test case set should be added if novel requirements exist. But for the situation that the position and the order of GUI change, only interface object recognition recording and mapping should be carried out again so that the original automatic testing set can be executed to achieve a prompt regression test.

4. Verification by Experiments

In this part, a mobile app of office automation (OA) system is tried out to validate the feasibility of the proposed method. In this OA system, a large number of approval processes are needed, as depicted in Fig. 5, and the process will adjust itself with different management requirement. During the requirement analysis and preliminary design stages of the testing and validation, the reusable scripts are designed for the GUI elements that the OA system used, such as Button_Click(name), List_Select(name), Text_input (name, text), DateTimePicker(name) and Text_Check (name, checkValue). For example, the operational GUI elements contain button, Check button, scroll bar and menu and the input GUI elements include textbox, list box, tree view, list view and combo box. Secondly, the reusable validation script, like Button_Check(name,checkValue) and Text_Check (name, checkValue), is established via the ways of test case expected result validation, such as textual value comparison, figure comparison. In detailed design and coding stage, testing engineers design test cases and developers script the test cases in parallel. At the end of the development, the OA system is submitted to be tested to construct the relationships between script and real GUI interface by recording the script and to execute the test.



CW-012 Borrowing/Ref More

* Borrowing/Refund
* Applicant
* Department
* Payee
* Reason
Expenditure Project Number
* Money(¥)
Loan Order Number

Submit

XZ-005: Meeting Room More

* Applicant	Chen JianQiang
* User Department	Marketing Depart
* Meeting Room	615
* Reason for Use	Reception
* Start Time	2019-01-18 09:30
* End Time	2019-01-18 10:30
Remarks	

Submit

XZ-008 Seal Applica More

* Applied Department
* Applicant
* Seal Applied
* Sended Department
Copies
* Reason for Seal
Interpretations
Picture

Submit

(1) Sealing Application Interface (2) Meeting Room Application Interface (3) Loan Application Interface

Figure 5. Tested Application GUI Interface

In Table 3 and Table 4, we can see that most of the reusable scripts can not only be directly applied and effectively called in different GUI testing for a single project since they are developed, but also be shared in across-project condition. During three-round regression testing, the modification rate of the test script and the interface object mapping script are less than 15% and will be going down with the regression testing. In addition, the reuse rate of the reusable script is more than 98%. With the proposed GUI automatic testing method using three-layer script framework, this OA system saves 37.4% time compared with that consumed by manual testing, implementing an unattended daily build and test and dramatically decreasing the cycle of the system testing of the whole project.

Table 3. Sharing rates for reusable script with different GUI interfaces

Tested Application GUI Interface	**Input/Operation Number**	**Quantity used (Reusable Script)**	**Sharing Number (Reusable Script)**	**Sharing Rate**
Sealing Application Interface for Sealing	7	4	4	100%
Meeting Room Application Interface	7	5	4	80%
Loan Application Interface	8	4	4	100%

Table 4. Modification rate and reusability rate for scripts with different regressive editions

Edition	**Modification Rate (Test Script)**	**Modification Rate (Mapping Script of Interface Object)**	**Reusability Rate (Reusable Script)**
Regressive Edition 1	14.73%	9.12%	98.2%
Regressive Edition 2	10.36%	6.2%	98.7%
Regressive Edition 3	5.85%	3.5%	99.1%

5. Conclusion

Automatic testing method for GUI based on three-layer script framework not only fuses the advantages of main-current test automation framework, such as the data-driven testing framework, the keyword-driven or table-driven testing framework, the test library architecture framework, the test script modularity framework. Also, it avoids their disadvantages. The universality and flexibility for testing script are implemented via constructing large amount of reusable script libraries and utilizing data-driven or GUI mapping script, which can dramatically reduce the number of testing scripts, workload for maintenance and probability of success for automatic testing. Since the software requirement is determined, the proposed method makes it possible for engineers to carry out test case design and automation test script design in parallel. Moreover, with the separation of test automation and test design, the advantages of different testers and designers are utilized to make full use of so that the resources are rationally assigned under the condition of a shorter cycle of the project.

References

[1]. X. Y. Ma, Q. Chen, Q. R. Si. Research on Testing Case Reuse Technology of Telemetry Software [J]. Modern Electronics Technique, 2015, 38(16): 29-33.

[2]. B. Lipika and T. Sanjeev, “GRAFT: Generic & Reusable Automation Framework for Agile Testing”, in IEEE 2014 5th International Conference-Confluence the Next Generation Information Technology Summit, edited by P. K. Singh. (Noida, India, 2014), pp. 761–766.

[3]. P. H. Jiangg, J. M. Xu. Web Application Testing Framework Based on MVC Model and Behavior Description [J]. Modern Electronics Technique, 2017, 40(6): 71-74.

[4]. R. N. Dang, J. Chen. Research and Implementation of Web Automation Framework Based on Keyword-driven [J]. Industrial Control Computer, 2017, 30(09): 46-47.

[5]. W. X. Zhang. Constructing Maintainable Script Technique in Software Testing Automation [J]. Electronic Technology & Software Engineering, 2017, (24):62.

[6]. J. Y. Yao. Analysis on Script Technique in Software Testing Automation [J]. China New Communication, 2018, 20(08):165-166.

[7]. K. Liu, X. Liang, J. P. Zhang. Research upon Software Testing Process Model [J]. Computer Science, 2018, 45(11A): 518-521.

[8]. Y. M. Yan. Research of Software Testing Process Model Based on Workflow [J]. Computer Engineering & Software, 2018, 39(05): 160-165.

[9]. J. Itkonen M. Mantyla, C. Lassenius. The Role of The Tester's Knowledge in Exploratory Software Testing [J]. IEEE Transactions on Software Engineering, 2013, 39(5): 707-724.

[10]. X. Mo, F. Zhao. Data-model-driven Software Automation Test Framework [J]. Computer Engineering, 2009, 35(21): 78-81.

[11]. J. J. Huang, Y. M. Li, J. Liu, H. Zhou. Design and Implementation of Automatic Testing System Based on Python [J]. Modern Electronics Technique, 2017, 40(04): 39-43.