Research and implementation of software automatic test

LIAN Li-hong
* Xiamen University Tan Kah Kee College, Zhangzhou Campus of Xiamen University, 363105, China

Abstract. With the fast development in IT technology nowadays, software is increasingly complex and large. Hundreds of people in the development team, thousands of modules and interfaces, across geographies and systems user are no longer a fresh thing. All of these put forward higher requirements for software testing. Due to the low cost of implementation and the advantage of effective inheritance and accumulation of test assets, software automation testing has gradually become one of the important means to ensure the quality of software for IT enterprises. This paper analyzes the advantages of automatic test, common misconceptions; puts forward unsuitable application scenarios and the best time to intervene; focus on the analysis of the feasibility of judging the interface automation test; and puts forward the function and elements of interface automatic test tools to have; provides a reference for large-scale project interface automated testing tool selection or custom development.

1 Introduction
Current software applications have penetrated into various fields, from domestic appliances and consumer electronics to precision instruments in national defense, aerospace and medical fields. As an important means to ensure software quality, software testing has run through the whole life cycle of software. With the rise of mobile internet, big data, cloud computing and other information technology, the traditional manual testing has encountered great challenges, such as: the system is large, more functional points, frequent changes in demand, the rapid update, etc. How to get the best test result with the least manpower becomes a difficult problem to be solved urgently in the software testing, so the software automation testing has been paid attention to and becomes the main trend of software testing.

2 Automatic test overview
2.1 Background and advantages
From the life cycle of software engineering, the testing work includes requirement analysis, case design and case execution.

Among them, the requirement analysis and case design have certain creativity, but case execution is relatively mechanized and repetitive. Therefore, there are a large number of automatic test tools for the mechanized operation in the implementation of the case. Currently the most influential automatic test tools on the market are QTP and Selenium[1-3].

Software automatic test has the advantages of high efficiency, convenience, time saving, and labour saving, etc. It has been widely used in the process of software project testing. As shown in the literature[4], the software automation test has the following advantages: test assets can be reused, saving the cost of the enterprise, improve the efficiency of the test, solve the difficult test, etc.
2.2 Misunderstanding
The advantages of automatic test, which is bound to bring high expectations of users, followed by a number of common misconceptions, such as:

1) Automatic test can completely replace manual test
As the automatic test just put a small amount of human resources in the process of testing, and automatic test tools will not fatigue, without considering whether to work overtime, will not cause the fluctuation of the test quality because of the emotional changes of the testers, so many users expect whether can completely replace the manual test with automatic test. However, the software test automation is not as long as the test cases can be, but a comprehensive work, and the test results are affected by many factors, including: the test items are suitable for automatic test, development and test automation scripts in the process of maintenance cost is too high.

2) Automatic test can detect a large number of defects
The success rate of automatic test is closely related to the maturity of the system, the higher the system maturity, the higher the success rate of the automated test, and vice versa. At the same time, the more defects of the system, the lower the degree of maturity of the system, the more the system is not suitable for automatic test, and the logic design of automatic test is completed by manual, so the defect is inevitable. So most of the new software defects should be found by manual test instead of automated testing, such as testing expert James Bach concluded that 85% of defects found by hand, and automated testing can only find a defect of 15%.

3) Automatic test can be completely out of manual intervention
At present, the coupling degree between the software system and the system, the module and the module is relatively high. When testing a system or a module, you often need to rely on other systems or modules. The ability of cross-system or cross-module test is the lack of the ability of automatic test tools, in addition, the automatic test tool itself does not have the imagination, therefore, when the system needs to think, human-computer interaction, user experience and other test tasks, automatic test tools are powerless. Therefore, it is expected that automatic test will be able to complete the idea of human intervention in the actual testing process is still not feasible.

2.3 Scene selection
Can be seen from the above, automatic test is not a panacea, only in the right circumstances to play the advantages of dynamic testing, automatic test should be used to avoid the following types of projects:

1) Short cycle project
Software automation test script design is the need for professional staff to spend a considerable amount of time to prepare and maintain, the cost of introducing automatic test to high customization and shorter cycles is often higher than the cost of manual testing, therefore, it is not recommended to automate the test.

2) Extremely complex business rules
The business rules of the project are more complex, and the testing process often needs to be accompanied by artificial logic judgment and irregular business processing, it is difficult to quantify the rules of all kinds of logic, so it is very difficult to introduce automatic test in this kind of project.

3) Beautiful, sound, user experience, ease of use, human-computer interaction and other higher requirements of the project
For this kind of project, there is not a uniform standard and quantitative standard for personal feelings. Therefore, the test results of this kind of projects are very poor. In the process of human-computer interaction, it is necessary to do a lot of manual intervention, and it is not suitable for automatic test.
4) Software development project has not yet finalized
In the process of project implementation, the project will be accompanied by constant demand change
and functional defects. If the introduction of automatic test at this stage will often encounter
automation design scenarios can not be executed, the design of the script to change with the
requirements of the continuous changes and maintenance issues, the need to spend more manpower
costs.

In short, not all of the scenarios are the introduction of automatic test, users need to analyse the
necessary feasibility in the testing process. Automatic test usually can only cover 70% of the cases and
scenarios, in most cases need to complement each other and manual testing. The best time to introduce
automatic test in software testing, first of all, it must be more stable in the system, and the larger the
project or the higher the degree of automation of the project can play a role in the test. In addition, the
management process system is more suitable for automatic test, and multimedia systems are not
suitable for.

3 Automated test cases

3.1 Automatic test architecture
The essence of automatic test is the process of decomposing a complete test project into a standardized
test atom; here we test this test atom as a test step. As shown in Figure 1, a software test project
generally can be divided into 3 parts from large to small particles:

![Figure 1: Automatic test architecture](image)

1) Test scenario: a set of test cases that can be continuously executed to verify whether the software
function is executed properly.

2) Test case: a set of test steps that are used to verify that the results of a particular condition match
the expected results.

3) Test procedure: the specific steps in the testing process are the smallest atomic units of software
testing.

4) Test object: at present, the test object is generally understood as interface element.

In the middle and late stages of a relatively stable software project, especially in the process of
regression testing, test execution can be said to be the comparison process of mechanical and
repetitive, thus the introduction of automated software testing tools in this stage, the test scenarios, test
cases, and test steps that are decomposed in advance are executed sequentially according to a certain
time and rules, and could achieve the same effect of manual test.

3.2 Automatic test architecture
In this paper, we take the common interface test project as an example, and describe the interface
testing process briefly, as shown in figure 2:

1) Message technology definition: Define the protocol used in the message interaction, the short and
long connection, the target address of the interactive system and the format of the parse message.
2) Message template design: The message definition template is used to edit the message format template according to the format which can be supported by the automated testing tool, which is the definition of the elements of the interface message, including the size of the message element, the data type, and so on.

3) Test case design: Write executable test cases. In the process of interface automatic test, we generally believe that the interface test message template is relative format template, thus the same message template filling test data of different test cases can be considered to be different.

4) Test scenario design: Edit the execution sequence of test cases. There is a certain dependency between test cases, example: When the automatic deduction in the bank card, first need to know what is the balance of bank card.

5) Business rule check: One of the main features of the interface test project is that there are more rules for checking the message elements. The same message principle often has different business check logic under different conditions. Example: poor credit customers can not handle loans, more than 500 thousand of customer deposits are known as bank quality customers, etc.

6) Execution result setting: according to the execution scenario of the software, the expected results of different scenarios and test cases are set.

7) Comparison of test results: execute test cases and perform manual tests to determine if the results are in line with expectations.

8) Test result record: record the results and defects, wait for repair.

According to the interface testing process, we can find the nature of the interface test project is constantly changing the message template, filling the corresponding test data, and repeating verification of mechanical work. It is possible to consider the use of automatic test tools to automate the execution and implementation of automatic comparison of results to replace manual testing. Besides, the function of automatic testing tools can effectively monitor the implementation process of automatic test, and back to the implementation of each test case, such as the implementation of the process of automatic screenshots, execution logs, the implementation of early warning, etc., to facilitate the testing personnel to test the defects and the problem of positioning. In the software interface test project, the use of automatic test is not only technically feasible, but also can effectively improve the efficiency of testing and test management.

Taking the Selenium[5] as an example, this paper gives a brief introduction to the interface automation testing process. Selenium framework has the following characteristics:

1) Through hot swap way to support HTTP, RPC, soap and other common communication protocols, and provide the conversion function between different communication protocols, so that the communication process seamless connection.

2) Provide a variety of interface testing methods, including a single interface test and multi interface testing, currently more common for a single interface test.

3) Support the user to achieve accurate results verification through custom methods.

4) The use case development threshold is low. The user only needs to use the interface data to fill in the format file to automatically generate the use case.
5) Support the custom configuration for different or the same domain names under the same interface, that is, only a simple modification of the test platform configuration can easily be used on different platforms.

![Interface automation test flow chart](image)

Figure 3: Interface automation test flow chart

Use case of interface automation test tool execution flow as shown in figure 3. As can be seen above, the specific implementation process of the interface automation test is as follows:

1) Determine whether a single case execution: Judge case is a single case or a complex case, if it is a single use case, automation framework will choose a single case execution process, otherwise the automation framework will choose the composite case execution process.

2) A single case implementation process is as follows:
   Read the case: select the implementation case
   Loading data: select the corresponding test data
   Call components: call interface automation components
   Test assertions: comparative analysis of the results

3) The implementation of complex case is as follows:
   Find a case: select the implementation of the composite case
   Analysis of the case: the decomposition of the composite case into a number of sequential implementation of a single use case, the process requires manual intervention analysis.
   Call component: according to the split order of the compound case, execute a single case flow, and analyze the result

It can be seen that the essence of interface automation testing is to use the test base control provided by itself, and then a more complex interface use case is divided into several single interface test cases, thereby reducing the overall difficulty of testing, improve the quality of interface test.

In general, a more complete interface automation testing tool should at least include the following:

- Data dictionary configuration: functions used to define and maintain interface message specifications. Message specification should include at least the message elements, communication protocols, the need to increase the function of the barrier, etc.
- Execution environment configuration: used to define the environment configuration function of the host machine and the target system.
- Perform case configuration: the process of decomposing the message interaction process into multiple reusable test cases and steps is often a kind of manual selection operation with data matching.
Perform task configuration: define the time, polling, and other functions for automatic test execution.

Perform results view: any automatic test tool should provide a view of the results of the implementation, and the function should provide a different dimension view, including execution time, order, project, module, the members of a sequence related to the test life cycle factors.

Execution tracking function: for the interface test, the most convenient way to track is to get the complete message of send messages and receive messages, a better user experience of automatic test tools not only to provide a complete message process information page, it should be provided from the perspective of user convenience to design the function.

Perform results turn defect function: for the case that the interface automation test results are not consistent with the expected results, the interface automation test tool should be able to provide detailed information about the execution of the test into the defect. Due to the complexity of the causes of defects, in the case of automatic test results are not the same, the results of the test for the defect is also required by the human audit to enter the treatment of defects.

Visible, interface automatic test tools are not universal. Although in theory, in the process of interface automation testing, as long as the message, scripts, data are in line with the conditions of automated testing, automatic test can be done unattended, unlimited polling, however, in the actual operation process, due to the uncertainty of the physical environment, the limitations of the design of the test case, the unexpected situation, and so on. In general, it is recommended that a small number of operators to carry out effective monitoring of the implementation process of the interface automatic test in the process of automatic test, which can ensure the effective operation of the interface automatic test. In the process of using the interface automatic test, test automation good projects are generally necessary to set up special automation test development engineer and automation operation engineer to continuously adjust and develop the interface automation message, script, data and environment. As a result, the idea that interface automatic test does not require the involvement of other people will not work.

4 Summary
Software automation testing has become an important test method of IT quality assurance, but not all software projects can get a good return. Only in the appropriate time, the rational use of automatic test will effectively improve the quality and efficiency of the test and reduce the artificial error rate in the test process. Therefore, in the actual operation of software testing, it is necessary to combine the characteristics of software projects, and constantly optimize the automatic test program to ensure the accuracy of the test results.

References
[1] LI YU, YU Shuang-mei, WANG Tian-sheng, MA Zhao-han. “QTP Based Enterprise Application Software Automation Testing Method”, Computer system application, volume 25, pp. 219-224, (2016).
[2] ZHANG Hai-mei. “Testing method of coal mine safety monitoring and control system based on QTP”, Industry and Mine Automation, volume41, pp. 108-110, (2015).
[3] HE Jin-ning, DU Wei-wei, GAO JING. “Exploration and practice of software automatic testing”, Foreign electronic measurement technology, volume35, pp. 1-4, (2016).
[4] Zhang Hui-lin, Li Wei, Tong Qiu-li, Liu Yu-jiang. “Design of integrated automated testing platform based on Selenium and TestNG”, Experimental Technology and Management, volume32, pp. 153-159, (2015).