Research on Graphic User Interface Automation Testing Technology Based on Cloud Platform

Xin Liu

China electric power research institute, Beijing, 100192, China

Abstract. As the soul of computer, software has more and more important influence on human beings. GUI (Graphical User Interface) technology is widely used because of its friendly interface and easy operation. It is an important part of software nowadays. No matter what kind of software, from requirement analysis to detailed code design stage, we need to use excellent testing technology to lay the foundation for the smooth development of software. Software testing technology has become an important branch affecting the development of software industry. Especially in the regression test conducted in the later stage of the test, it is necessary to verify whether the previously discovered problem is solved in the new version. Automating software testing can make a large number of tests programmatically repeated. Not only does it save a lot of labor, but it also improves test efficiency and guarantees the quality of the test. This paper proposes a graphical user interface automation test technology for cloud platforms, which provides assurance for testers to easily create and flexibly maintain test code.

1. Introduction

As the soul of computer, software has more and more important impact on human beings. At the same time, many accidents make people more and more worried about the quality and safety of software products [1]. Testing is an activity designed to assess the properties or capabilities of a program or system and determine whether it meets its desired results. GUI technology is widely used because of its friendly interface and easy operation [2]. It is an important part of software nowadays. In the whole software development process, from requirement analysis to system design to code implementation, there will be more or less problems. Today, with the rapid development of industry, the rapid development of hardware has left a deep impression on people. Relatively speaking, software develops equally rapidly in terms of quantity. Large-scale system software and millions of lines of application software have become sparse and common [3]. The purpose of software testing is to ensure the quality of software development. Regardless of the software, from the requirements analysis to the detailed design code phase, it is necessary to use excellent testing techniques to lay the foundation for the smooth development of the software [4]. By studying a new cloud platform GUI automated test platform, it does not involve the modification of the code, and at the same time completes the real-time tracking of the cloud platform running process [5]. It can realize non-intrusive automated testing of cloud platform GUI, avoiding the bottleneck of existing cloud platform GUI automated testing technology [6]. Cloud platform GUI automation test status Cloud platform GUI automation test and common GUI.

Automated testing technology is an important research field in modern software engineering and an important means to improve the efficiency and reliability of software testing [7]. The development of software testing technology lags far behind many foreign countries. But with the rise of the software industry in these years, the country is also beginning to pay attention to software testing. Automated testing can shorten the software development life cycle, improve the speed of detection, performance
and security, and improve the accuracy and accuracy of software testing [8]. Software testing is a lot of work and has some repetitiveness. Software testing technology has received more and more attention from the current international software industry. Software testing technology has become an important branch affecting the development of the software industry [9]. Especially in the regression test conducted at the later stage of the test, it is necessary to verify whether the previously discovered problems are solved in the new version. The application scope of software is more and more extensive, and the structure is more and more complicated. Faced with such challenges, many software development process methods have been proposed [10]. Trying to ensure its quality through strict research and development process. Although the workload of software testing is very heavy, the reusability of test cases is very high [11]. When entering the later regression testing stage, it is necessary to find out whether the previous problems of the software have been solved or whether the software code has been changed to meet the previous requirements [12]. Software testing has become the focus and difficulty of current software technology research. Research on software testing technology and research and development of testing tools have been paid more and more attention by the software industry.

Automatic test technology is to simulate manual test steps, and execute test scripts written in language to automatically test software according to the predetermined plan of the tester [13]. Realizing the automation of software testing can make a large number of tests procedurally and repeatedly executed. It not only saves a lot of labor, but also improves the testing efficiency and ensures the testing quality. In order to ensure the quality of software products, software testing, especially automated testing, has received more and more attention [14]. With the deepening of software application, people's demand for software is increasing, and the functions expected to be provided are becoming more and more complex [15]. A test script is a collection of data and instructions written in a scripting language with a regular grammar, which is actually a programming language [16]. The development of software testing technology still can not meet people's expectations, and testing the software quality is a very practical topic [17]. In order to reduce the development cost of software products and shorten the software development cycle to ensure the quality of software products, we must pay attention to and strengthen the software automation test [18]. In order to ensure the quality of the software product itself, it is necessary not only to strictly control the software development process, but also to test the product itself [19]. This paper proposes a graphical user interface automation test technology for cloud platforms, which provides assurance for testers to easily create and flexibly maintain test code.

2. Graphic User Interface Testing Method

In order to ensure the quality of the test, the test code must have strong adaptability to the changing system. In the actual project development process, software testers are involved in the project very early. In the early stage of the project, the tester's intervention is to better carry out the mid-and late-stage testing work and to evaluate the feasibility of the project. Then the arrangement and utilization of test resources are estimated to minimize the development cost. At present, the development of software is mainly carried out by parallel development of different modules, which requires sufficient communication and communication between developers [20]. In the current software testing, one of the most concerned problems is how to realize the automated testing of GUI efficiently and make the testing code flexible. The correctness of the GUI is critical to the security, stability and availability of the entire software. In the middle and later stages of the project, the focus of the entire project will be transferred to the testers. To ensure the quality of software products, testing is the most important means. Testing of the GUI is very important, and the success of the GUI test will affect the overall quality of the software product.

Interactive software is a typical GUI software whose execution state and result are determined by its initial state, the data and operations entered during execution, and the chronological order in which it occurs. During the software development cycle, the system needs to be constantly updated and maintained [21]. The execution process of the interactive software is shown in Fig. 1.
The script operation module parses, classifies, and determines what type of event is the content of the event. Modifying the code is inevitable because we always have to correct the error and the requirements are subject to change. A vertex with no precursor vertices in the interface component association graph is called the starting vertex, and a vertex in the associated graph without subsequent vertices is called the final vertex. Fig. 2 is an interface member association diagram in which "v1" is the starting vertex and "v8" is the final vertex.

The test method for constructing the association graph based on the interface is to establish an interface component association graph according to the software requirements specification. Describe the interconnection and restriction relationship between each component in the interface and other interface components. For testers, the most important is the documentation that is generated during the project phase. When the GUI event capture module calls the write driver interface, the captured event content is passed to the script operation module. Reliability and sufficiency depend on the extent to which the test case covers the interface component association graph. That is, the higher the degree of coverage, the higher the reliability and sufficiency of the tested GUI software. After the automation object is located, the message sending module provides a corresponding interface to send a message to the system [22]. When the system responds, it notifies the application to trigger the automation object to execute the corresponding event. Options for automated testing tools need to be listed, which is very helpful for project managers to allocate resources reasonably. Starting software development without detailed and adequate communication often leads to too many functional errors in the whole system at run time.

Node accessibility plays an important role in the analysis of communication channels and anti-interference of network traffic. Because the data center network is relatively closed and centralized, it is driven by different performance requirements in the actual deployment process, showing the coexistence of a variety of network forms. It conforms to the premise of cointegration test. Test whether there is a long-term equilibrium relationship between the relevant variables. The results are shown in Table 1.

### Table 1. Cointegration test results

| Eigenvalues | Trace estimate | Threshold |
|-------------|----------------|-----------|
| 0.741       | 1007           | 138.7     |
| 0.614       | 954            | 141.2     |
The development of software projects is a complex and mutually assisted process, and there are many unexpected problems in the whole process of project development [24]. Often solving these problems will cost developers too much time. In the cloud platform model conceptual model, the cloud platform schema entity refers to an objective object that can engage in the cloud platform model. Fig. 3 is a conceptual model of a cloud platform model and a graphical user interface system.

![Conceptual model of cloud platform model and graphical user interface system](image)

According to specific functions, the management module layer is divided into a monitoring module, an algorithm module, and a control module. In a network with i nodes, the normalized tightness index P is defined as:

$$P = P(Y = 1) = F(\beta_iX_i)$$  \hspace{1cm} (1)

The data point is defined as the minimum of the distance from point $\theta(P)$ to any point with a higher local density point:

$$\theta(P) = \text{Logit}(P) = \ln \left( \frac{P}{1 - P} \right)$$  \hspace{1cm} (2)

The inertia weight still decreases linearly with the increase of the number of iterations, which effectively improves the search accuracy and convergence ability. The update formula is as follows:

$$y_{f-n_a} = \sum_{i=1}^{N} \sum_{j=1}^{M} \sqrt{P_i h_{i,n_a}^T W_{i,a} s_i}$$  \hspace{1cm} (3)

Some resources are in a state of high load for a long time, and some resources are always idle. Calculate the average of each cycle as shown in the formula:

$$P_{f-n_a} = \sum_{i=1}^{N} \sum_{j=1}^{M} P_i \left[ h_{i,n_a}^T W_{i,a} \right]$$  \hspace{1cm} (4)

If the transmission line is converted to a telephone line during the transition to a telephone line, it can be used by converting it into an analog signal and then converting it into a data signal. The best case for all nodes to be fully concurrent and without network competition. In the worst case where all node communication sequences occur, the number of equivalent serial communications is equal to the sum of the actual number of communications in each node. Fig. 4 is an example of communication in a three-dimensional network.
The cloud platform software graphical user interface automated test platform directly collects the video data output by the system under test to the display without disturbing the system under test. As software interfaces become more and more complex, software testing has become a complicated and cumbersome task, posing great challenges for testers. During the batch automation run of test cases, logging not only needs to include execution details of the test cases. It also provides information on the environment, time, number, and status of test case execution [23]. Testers should document defects in defect management tools and help developers reproduce those defects. Testers need regression testing to track these issues, and if these issues are identified as being modified, the tester can close the defect. We need to extract events for software components from a software perspective. Then all events are grouped into event queues. Finally, the system executes the event queue in order to complete the regression test. As long as the object being manipulated exists in the user interface, even if the position or image resolution changes, the test code will locate and operate accurately according to the characteristics of the object.

3. Automated Testing Method Based on Interface Component

If automated testing tools are introduced into the testing process without a formal process, the results can make the testing project messy and non-reusable. There are many changes in project development, most of which are brought about by changes in requirements. Customers usually do not understand the way software development is done, nor do they know their actual needs. It is often after a certain period of product development that users understand their real needs. Software testing is the last pass in software quality system. The quality of software testing determines the fate of success or failure of the software. So in order to ensure the quality of software, we must first improve the software testing technology. Some tests on software performance require a high level of execution, and can not be tested without the help of automated tools. If the repeated changes in demand, it is easy to affect the enthusiasm of the software developers, the work can not achieve a sense of accomplishment. This makes the developer's work interest less, and it is easy to lay aside defects in software development. Automated regression testing software can replace manual input test cases by running event queues, but it is also the source of its lack of flexibility. According to the execution steps of the test case, the human-computer interaction step is converted into a control command of the mouse and keyboard simulation device, and the control of the tested software mouse and keyboard is realized. Thus, the required human-computer interaction information is input to implement the execution of the test case.

Automation customers can create automation objects, access objects provided by the automation server, get or set properties of objects, or call methods on objects. The interaction between the automation object and the automation customer is shown in Fig. 5.
Since the test belongs to the black box test category, its test mainly focuses on the functional aspect, and needs to verify the accuracy of the test through the results displayed by the interface. Based on the existing GUI test theory and model, from the perspective of logic function, a GUI automation test method based on interface component is proposed. The event flow of the user operating GUI software is decomposed into components with a limited number of interface states and interface functions, and new interface components can be derived by establishing constraint relationships between components. Test coverage criteria are presented for potential test points using elements in the interface component structure diagram [25]. When an unexpected situation arises, the test software can not eliminate the interference intelligently and timely as human beings, and it will still be executed step by step according to the designed test script. Functional testing of each component in the interface is carried out to check whether the function of the component meets the requirement analysis criteria of the software system. Combine functional testing with interfacial testing to complete the test.

A node with a large compactness may not have a large degree value, but it plays a very important role in data transmission. The degree of some nodes is not very large, but their parameters are relatively large. It reflects the core degree of these nodes in the network. If these nodes are deliberately attacked, the whole network may be paralyzed. For example, Table 2 shows the data between nodes, tightness and parameters. Fig. 6 shows the relationship among node degree, compactness and parameters.

Table 2. Node degree, tightness, and data between parameters

| Node degree | Tightness | Node parameter |
|-------------|-----------|----------------|
| 1           | 0.86      | 0.42           |
| 0.68        | 1         | 0.67           |
| 0.74        | 0.69      | 1              |
Fig. 6 Relationship between node degree, tightness, and parameters

Automated testing minimizes the overhead of software development and allows for more testing tasks in less time. Software may discover dependencies between modules during development, and these dependencies may affect each other, causing more problems. Cloud platforms carry more and more traffic. Research on the security and stability of cloud platforms is becoming more and more important. Fig. 7 shows the relationship between the name node size and the data node size.

Fig. 7 Name node size and data node size relationship

The final result is generated based on the situation at runtime or the parameters entered by the user. Calculate the actual output of each unit of the output layer:

$$cell_{mu} = \arg\max_u \left( \sum_{n=1}^{M} P_{f-nu} \right)$$

(5)

The server will describe the file based on the release of the application. Fix connection weights and thresholds:

$$cell_{mu} = \arg\max_{n,\mu \in R} \left( \sum_{n=1}^{M} P_{f-nu} \right)$$

(6)

When the display format is important, you can specify how to display the data. Finding the optimal solution set makes each objective function as large or small as possible. In the actual application process, the algorithm is as follows:

$$RSRP_{n,\mu} + (\lambda_j - \lambda_k) = RSRP_{n,\mu}$$

(7)

When you need to establish a database connection in your program, you only need to take one from the memory instead of rebuilding a connection. Discuss this issue, and the mutual formula is as follows:
The traditional communication identification IP address not only indicates the location information of the application service, but also indicates the identity information of the application service, which makes the isolation measure greatly restrict the flexibility of resource scheduling, resulting in the generally low usage rate of network resources. A process of promotion, application, and upgrade is needed. Different sounds will appear in this process. This requires the construction department to obtain support from the decision-making level. Fig. 8 is the equalizer architecture model.

\[
M(t) = R_0 (1 + \frac{2\Delta R U}{Q_0 R_0^2} t)^{0.5}
\]  

Fig. 8 Equalizer architecture model

Event sets are composed of a limited number of orderly interface operations, which can be divided into system operations and user operations. The state is the static description of the component, and the target state of other interface components can be used as the starting state of another interface component. Any control, even the simplest one, may contain very complex programs. After configuring and running data analysis, the number of information nodes used in each analysis is different, and the corresponding processing time is also different. Table 3 shows the number of nodes and processing time for each analysis. The relationship between the number of nodes and the processing time is shown in Fig. 9.

| Serial number | Number of nodes | Processing time |
|---------------|-----------------|-----------------|
| 1             | 4520            | 14812           |
| 2             | 4618            | 19741           |
| 3             | 4732            | 16359           |
Fig. 9 Number of nodes and processing time

In a network with \( h \) nodes, the sum of the distances of nodes to other nodes is generally not lower than \( h-1 \), then the normalized compactness index \( P_h \) is defined as:

\[
P_h = \frac{\sum_{i=1}^{h} \sigma_i}{\sum_{i=1}^{h} \sigma_i^2}
\]

(9)

A data point is defined as the minimum of the distance from point \( k \) to any point with a higher local density point:

\[
U_{ij} = \frac{H_{ij}}{\sqrt{\sum_{i=1}^{k} H_{ij}^2}}, i = 1,...,n, j = 1,...,k
\]

(10)

The particles will update their speed and position according to the formula below:

\[
I(X;Y) = \sum_{i,j} \rho(x,y) \log(\frac{p(x,y)}{p_i(x)p_j(y)})
\]

(11)

A GUI test case consists of a sequence of events run by the driver, called an event stream or a test event stream. When the GUI event is played back, the target program is displayed at the front end of the screen. The playback module then notifies the script module to load the script file, sequentially reads the various GUI event nodes in the script and parses the various GUI event parameters. An interface component is an ordered combination of a set of event sets and interface state sets. The more user-friendly the user interface, the more complex the code behind it may be. With the increasing complexity of software development, the importance of software testing has become increasingly apparent, and the development of software testing technology has also received more attention. A test code can run successfully in a multilingual environment, perfectly realizing the cross-linguistic ability of automated testing. Traditional GUI testing methods use recording/playback tools to record and generate linear scripts containing sequence of running events. Although this method can generate event execution sequence directly and effectively, the test data and event operation sequence are immobilized.

4. Conclusion

Automated testing framework is undoubtedly an inevitable development direction for enterprises to implement automated testing. It is important for the proper foundation of successful automation in production. As the current software structure is becoming more and more complex and the scale is becoming larger and larger, manual testing has been unable to meet the needs of testing of course. Therefore, test automation has become the main development trend of software testing. Automated testing does not require human intervention, and the test of software products is realized through the automatic execution of scripts. The purpose of automated testing is to help manual testing, improve
testing efficiency and reduce testing costs. Although communication and synchronization between multi-machine processes are currently implemented, no connection has been established with the test case manager. From a practical point of view, the system does not require complicated calculation steps and does not require the user to carry additional authentication equipment. There are many situations in the GUI test process that need to be compared with the echo information under the command line. These echo information may be very complicated or irregular. It is hoped that the automated test tool can provide a simple way to process the information of the echo when it is provided to interact with the command line to make the use case of the interactive test clearer.

References
[1] Nguyen B N, Robbins B, Banerjee I, et al. GUITAR: an innovative tool for automated testing of GUI-driven software[J]. Automated Software Engineering, 2014, 21(1):65-105.
[2] Lin Y D, Chu T H, Yu S C, et al. Improving the Accuracy of Automated GUI Testing for Embedded Systems[J]. IEEE Software, 2014, 31(1):39-45.
[3] Amalfitano D, Fasolino A R, Tramontana P, et al. MobiGUITAR: Automated Model-Based Testing of Mobile Apps[J]. IEEE Software, 2015, 32(5):53-59.
[4] Memon A M, Cohen M B. Automated testing of GUI applications: Models, tools, and controlling flakiness[J]. IEEE, 2013, 8104:1479-1480.
[5] Strazza C, Del Borghi A, Gallo M. Development of Specific Rules for the Application of Life Cycle Assessment to Carbon Capture and Storage[J]. Energies, 2013, 6(3):1250-1265.
[6] Macik M, Cerny T, Slavik P. Context-sensitive, cross-platform user interface generation[J]. Journal on Multimodal User Interfaces, 2014, 8(2):217-229.
[7] Lin Y D, Rojas J F, Chu T H, et al. On the Accuracy, Efficiency, and Reusability of Automated Test Oracles for Android Devices[J]. IEEE Transactions on Software Engineering, 2014, 40(10):957-970.
[8] Schilders L, Daoudagh S, Lonetti F, et al. Automated testing of eXtensible Access Control Markup Language-based access control systems[J]. IET Software, 2013, 7(4):203-212.
[9] Gebremariam S Y, Martin J F, Demarchi C, et al. A comprehensive approach to evaluating watershed models for predicting river flow regimes critical to downstream ecosystem services[J]. Environmental Modelling & Software, 2014, 61:121-134.
[10] Harte S E, Mitra M, Ichesco E A, et al. Development and validation of a pressure-type automated quantitative sensory testing system for point-of-care pain assessment[J]. Medical & Biological Engineering & Computing, 2013, 51(6):633-644.
[11] Sun C A, Zhao Y, Pan L, et al. Automated Testing of WS-BPEL Service Compositions: A Scenario-Oriented Approach[J]. IEEE Transactions on Services Computing, 2018, 11(4):616-629.
[12] He W, Zhao R, Zhu Q. Integrating Evolutionary Testing with Reinforcement Learning for Automated Test Generation of Object-Oriented Software[J]. Chinese Journal of Electronics, 2015, 24(1):38-45.
[13] Ratiu D, Voelter M. Automated Testing of DSL Implementations - Experiences from Building mbeddr[J]. Software Quality Journal, 2017(3):1-36.
[14] Miller N, Gruson D. Implementation of automated testing for 1,25-dihydroxyvitamin D: Return of experience from a core-laboratory[J]. Clinical biochemistry, 2015, 49(3):298-301.
[15] Thornton C, Cohen O, Denzinger J, et al. Automated Testing of Physical Security: Red Teaming Through Machine Learning[J]. Computational Intelligence, 2015, 31(3):465-497.
[16] Suzuki K, Shimamori T, Sato A, et al. Detecting endotoxin activity in bovine serum using an automated testing system[J]. Journal of Veterinary Medical Science, 2015, 77(8):977-979.
[17] Szklarski P C, Booth G S. Horton sees a speck: automated testing undone by a fragment[J]. Transfusion, 2018, 58(3):620-621.
[18] Khari M, Kumar P, Burgos D, et al. Optimized test suites for automated testing using different optimization techniques[J]. Soft Computing, 2017, In press(In press):1-12.
[19] Pham R, Kiesling S, Singer L, et al. Onboarding inexperienced developers: struggles and perceptions regarding automated testing[J]. Software Quality Journal, 2016, 25(4):1-30.

[20] Berzins V, Van Bentheim P, Johnson C, et al. Use of Automated Testing to Facilitate Affordable Design of Military Systems[J]. Journal of Cognitive Neuroscience, 2015, 22(6):1095-1111.

[21] Bogdanov K, Maguire G Q. Toward Automated Testing of Geo-Distributed Replica Selection Algorithms[J]. Acm Sigcomm Computer Communication Review, 2015, 45(5):89-90.

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

[23] Jee E, Shin D, Cha S, et al. Automated test case generation for FBD programs implementing reactor protection system software[J]. Software Testing, Verification and Reliability, 2014, 24(8):608-628.

[24] Forest F, Rashed A, Huselstein J J, et al. Fast power cycling protocols implemented in an automated test bench dedicated to IGBT module ageing[J]. Microelectronics Reliability, 2015, 55(1):81-92.

[25] Xu D, Xu W, Kent M, et al. An Automated Test Generation Technique for Software Quality Assurance[J]. IEEE Transactions on Reliability, 2015, 64(1):247-268.