Research on the Effectiveness Design of Information System Software Test Case Based on Clustering Algorithms

Zeng Jijun\textsuperscript{1,2,*}, Mei Famao\textsuperscript{1,2}, Lin Jiaxin\textsuperscript{1,2}

\textsuperscript{1} Information Center of Guangdong Power Grid Co. Ltd, Guangzhou 510600, China
\textsuperscript{2} CSG-key Laboratory of Information Technology Testing, Guangzhou 510600, China
*Corresponding Author email: 915647780@qq.com

Abstract. With the rapid development of software industry, the scale of software becomes larger and larger, and the complexity of software becomes higher and higher. The test coverage criterion traverses to get the corresponding test path and deletes the redundant path according to the constraints, then analyses and eliminates the contradictions in each test scenario to get the consistent requirement scenario, and finally generates the test case set of interlocking software. Test cases are the basis of test execution. Effective writing of test cases plays an important role in the whole testing process of information system. There are constraints (such as dependencies and mutual exclusions) between input items or parameter values, and the constraints certainly exist. From the initial frequent pattern mining to closed pattern mining, maximum pattern mining, incremental mining, topic interest measurement, privacy protection, data flow and other types of data mining association rules.

1. Introduction

Traditional statistical analysis is usually more direct, because there are a series of clear expected results [1]. Clustering analysis technology is combined with adaptive random testing, and wavelet transform is improved. A measurement mechanism of test case distance is proposed. By using this mechanism, the test method of combining clustering analysis technology with adaptive random testing is applied to the test of information system software [2]. It plays an active guiding role in developing testing activities, finding and solving software defects as soon as possible, reducing software development risks and costs, and ensuring software quality. That is to say, the verification of external and internal quality of software is emphasized, while the quality of service that reflects the final performance of equipment software, that is, the user's demand for use, has not been given much attention [3]. As the core of the whole process of software testing and testing, test cases are the basic basis for testing execution. One of the biggest features of the test method is to execute the tested source program during the test. The basic idea is to make the program controllable and observe the behavior of the program from various angles [4]. Therefore, dynamic testing must include the program under test and the data used to run the software. Tests for input-output causality, tests based on business data equivalence classes, process-based tests [5].

In the stage of requirement analysis, system requirements are captured by use cases, the relationships between classes are described, and the collaborative relationships between use cases and classes are described by dynamic models [6]. In the coding phase, classes are represented by object-oriented programming languages. Determine the number of input parameters, the number of values of each parameter, and then select the appropriate orthogonal table for table head design [7].
The uniformity of coverage distribution in pairs and combinations of parameters is better than that in single-parameter rotation. It is very difficult to locate and correct each error, and at the same time, it may introduce new errors. It is more difficult to determine the cause and location of the errors because of the mixture of old and new errors [8-10]. The other is the incremental integration method, which expands the program step by step and enlarges the scope of testing step by step [11]. Write test cases during the development phase, refer to the requirements specification and software function points to refine the operation of each function point, and tend to reach the maximum demand coverage as much as possible [12].

In summary, our contributions are as follows: 1. This algorithm is a new clustering algorithm for the problem of validity design of information system software test cases. 2. This algorithm is widely applicable in the clustering environment, and it has high applicability for the validity design of information system software test cases. 3. This algorithm is more efficient, accurate and fast.

2. Materials And Methods

In the software testing of information system, the quality of software testing cannot be guaranteed because of the emphasis on operation, management, result and process. Each test case is associated with the test type, and each functional test case needs to cover all test types, such as portability, interoperability, stability and so on. The coding phase can perform unit and component level test execution according to the written test cases. The software cycle phase is shown in Table 1 and Figure 1. The test configuration management, test process supervision and test process review jointly guarantee the quality of the test. According to the test level, it is divided into unit test, integration test, configuration item test and system test.

| Table 1 Software Cycle Phase |
|-----------------------------|
| Demand | Test |
| Definition phase | 0.61 | 1.05 |
| Development stage | 0.53 | 0.93 |
| Maintenance phase | 0.72 | 0.81 |

Figure 1 Software Cycle Phase

In the process of test execution, it is certain that some test paths or data are not reflected in the use cases. How to evaluate the validity of test cases plays an important role in the test process. Information system software testing process should be modeled as software development process. In practice, software testing often does not have a certain process, such as testing without planning, random. The probabilistic rule algorithm is more stable, so long as the support and confidence are given, the deterministic rules can be obtained. In data space, the query method is generally designated keyword query, and the semantic expression ability of this query method is not strong. The uncertainties include
the content, transformation process and query results described by the main keywords. However, there may be multiple rules covering the same object, resulting in the number of rules is not concise enough. Testing should be accompanied by the whole software development life cycle, and the test object should include program, document and design, so as to facilitate the early detection and resolution of errors, reduce the risk and cost of software development, and ensure the quality of software. During the test execution, the input field is divided into four parts, and then a test case is randomly generated from the largest sub-domain. If the test case fails to detect an error, it can not only describe the interaction between the objects, but also describe The interaction between any other entity. Improved definitions of modeling concepts such as conditions, time constraints, object creation, lifeline termination, self-calling, and operational overlap. A series of new concepts such as frames, interactive segments, combined segments, and interactive use have been added. Then continue to divide the subdomain into sections to generate the next test case. For software product developers, the technology used to ensure software quality is mainly software review and software testing. Software review has its unique advantages over software testing.

In the testing phase, different test types are used for different levels of testing. The corresponding state of software testing phase is shown in Table 2 and Figure 2. The ratio of each test sequence and its test cases is generally not 1:1, and needs to be analyzed according to the specific situation. The basic set test is shown in Table 3 and the Figure 3.

| Table 2 Corresponding State in Software Testing Stage |
|------------------------------------------------------|
| **Operation** | **State** |
| Unit testing | 1.36 | 0.72 |
| Functional testing | 0.95 | 0.63 |

Figure 2 Corresponding State in Software Testing Stage

| Table 3 Basic Set Testing |
|---------------------------|
| **Interactive** | **Cycle** |
| Basic data set | 16.73 | 15.80 |
| Number of base users | 15.07 | 14.93 |
| User same operations | 15.22 | 14.31 |
The requirements for testing activities in the development process of command information system are divided into four categories: unit, component, configuration and system. Testing activities are divided into test requirement analysis and planning, test design, test execution and test summary. Push the transition and target node into the stack and mark the path as accessed, and record the target node as the current node. If the set of edge regions is empty, the decision-making system is compatible, testing each module, function, level and type. Detailed functional test cases are key modules, which are prone to errors, and deterministic rules are extracted. On the contrary, incompatible systems and uncertain rules are extracted. The general processing method is to first construct an input constraint relationship table, and then use the preferred use case to check the table to determine whether there is a constraint relationship. If there is a deletion, the use case is not executed. The movement is constrained by the postcondition of the source node, the guard condition of the edge, and the precondition of the target node. When the condition is met, the movement can occur, unlike the action in the state being executed, it takes time to fire.

If the attributes in information system are divided into conditional attributes and decision attributes, then the information system is called a decision information system. The decision parameters are shown in Table 4 and Figure 4.

| Table 4 Decision parameters |
|-----------------------------|
| Policy decision | Optimization |
|-----------------|--------------|
| Object set      | 6.71         | 5.13         |
| Property set    | 3.98         | 6.52         |

Figure 3 Basic Set Testing

Figure 4 Decision parameters
3. Result Analysis and Discussion

It needs to deal with unstructured, complex semantic structure and distributed stored data. Objects are randomly selected from the data set as initial representative objects. Then the distance between each remaining object and the representative object of each cluster is calculated, and each object is assigned to the nearest cluster. Then a candidate representative object is found in each cluster, and the distance between the candidate representative object and all the objects in the same cluster is calculated. Selecting suitable components from the organizational component library according to the system component assembly scheme is a special sub-stage of component-based development. If the set of attribute values is not empty, then the remaining attribute values are intersected to generate a new association rule precursor. By analogy, when the set of attribute values is empty, the algorithm ends. This method is based on the set of attribute values, and generates eligible association rules after intersecting the set of attribute values. The expected results should be clearly described to ensure that the tester can compare the actual results of the test. Activities and tasks and corresponding responsibilities are assigned to key participant managers, developers and testers, users or customers during the testing process. Several people with different responsibilities fulfill their respective responsibilities and achieve the predetermined sub-goals by completing various tasks and activities related to the sub-goals. Each function node substitutes input items and data of related functions, and sets conditions for branch, loop, parallel, and selection to form a test case set.

The accuracy of the scheme for determining the number of clusters is tested. The scheme is tested on different spatial point sets because their ideal clustering numbers are known beforehand. Moreover, the standards for these are also simple, so their segmentation results are direct, the optimal cluster number is determined as shown in Table 5, and the optimal clustering is shown in Figure 5.

| Table 5 Clustering results |
|-----------------------------|
| Implement | Mean value |
| Cluster 1 | 275 | 45 |
| Cluster 2 | 450 | 41 |
| Cluster 3 | 371 | 43 |

4. Conclusion

In this paper, the validity design of information system software test cases based on clustering algorithm is studied. The required functions and performances are relatively complete, but it still needs to be maintained and improved by system maintainers in the process of subsequent information system use, so as to make it play a greater role. With the understanding of software testing of information system, the test management information system of testing station has been formed, and the software
testing technology has been successfully applied to practical testing work. Testing methods have a certain randomness in selecting test cases to execute, and it is very likely that similar test inputs will be repeated, which results in similar results and reduces the effect of defect detection.

References
[1] Nanda S J, Panda G. Design of computationally efficient density-based clustering algorithms[J]. Data & Knowledge Engineering, 2015, 95:23-38.
[2] Ding S, Jia H, Zhang L, et al. Research of semi-supervised spectral clustering algorithm based on pairwise constraints[J]. Neural Computing and Applications, 2014, 24(1):211-219.
[3] Chawla, Suruchi. A novel approach of cluster based optimal ranking of clicked URLs using genetic algorithm for effective personalized web search[J]. Applied Soft Computing, 2016, 46:90-103.
[4] Zhou Z Q, Xiang S, Chen T. Metamorphic Testing for Software Quality Assessment: A Study of Search Engines[J]. IEEE Transactions on Software Engineering, 2015, 42(3):1-1.
[5] Niros A D, Tsekouras G E, Tsolakis D, et al. Hierarchical Fuzzy Clustering in Conjunction with Particle Swarm Optimization to Efficiently Design RBF Neural Networks[J]. Journal of Intelligent & Robotic Systems, 2015, 78(1):105-125.
[6] Bringmann K, Friedrich T. Convergence of Hypervolume-Based Archiving Algorithms[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(5):643-657.
[7] Hassani M, Kim Y, Choi S, et al. Subspace clustering of data streams: new algorithms and effective evaluation measures[J]. Journal of Intelligent Information Systems, 2015, 45(3):319-335.
[8] Garousi V, Kucuk B, Felderer M. What We Know About Smells in Software Test Code[J]. IEEE Software, 2018:1-1.
[9] Garousi V, Felderer M, Hacaloglu T. What We Know about Software Test Maturity and Test Process Improvement[J]. IEEE Software, 2017, 35(1):84-92.
[10] Husinger F, S. François, Jerraya A A. Definition of a systematic method for the generation of software test programs allowing the functional verification of system on chip (SoC)[J]. Journal of Materials Chemistry B, 2018, 6(14):11-16.
[11] Ailem M, Role F, Nadif M. Model-based co-clustering for the effective handling of sparse data[J]. Pattern Recognition, 2017, 72:108-122.
[12] Thomas J, Chaudhari N S. Design of efficient packing system using genetic algorithm based on hyper heuristic approach[J]. Advances in Engineering Software, 2014, 73:45-52.