Chapter 5　Black-box test design

5.1　Overview of black-box test

黑盒测试概述

Black-box test refers to the approach of treating the software under test as a black box that can’t be opened, and the internal logical structure and features of the software are not considered.

According to the specification of the software, the black-box test runs the software, inputs the test data, and checks whether the running results meet the specification.

Black-box test is a kind of test based on specifications from the point of view of users. Black-box test is also known as data-driven test.

5.1.1　Characteristics of black-box test

黑盒测试的特点

Black-box test, which focuses on the execution results and external characteristics of the software without considering its internal structure and implementation details, is commonly used for test the software as a whole, such as system test and acceptance test (as shown in Figure 5-1).

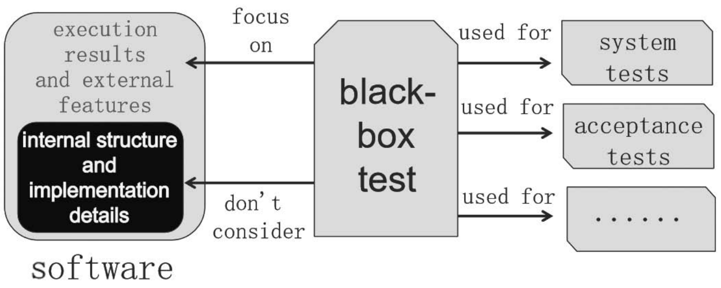


Figure 5-1　Characteristics and uses of black-box test

The design of black-box test cases can be conducted concurrently with software requirements analysis and design, thereby reduces the time required for the entire software project. For instance, during software requirements analysis, preparation for acceptance test and black-box test cases design for acceptance test can be carried out. Similarly, during system specification determining, preparation for system test and black-box test cases design for system test can also be carried out.

The primary foundation of black-box test is software specification. Therefore, prior to initiating black-box test, it is essential to ensure that the software specification undergoes a thorough review and meets the established quality requirements. If there is no specification, exploratory test can be adopted.

Black-box test can be used not only to test the function of software, but also to test the non-functional characteristics of software, such as performance, security, etc. For example, in preparation for the 2022 Winter Olympics in China, Shandong Inspur Ultra HD video industry Co., LTD dedicated 10 months to extensive research and successfully overcame obstacles in 8K Ultra HD live broadcasting technology. This enabled them to offer 8K Ultra HD decoder and video services for live broadcasting of the opening ceremony of the Winter Olympics.

8K is a kind of ultra-high-definition resolution, but the current domestic urban outdoor large screen only supports 4K resolution. In order to ensure the visual experience, the technicians decoded the video into four 4K signals, and then seamlessly spliced and synchronized the four parts of the picture.

Before the opening ceremony, technicians conducted extensive tests on large-screen circuits and control software, striving for that four images look almost perfectly in sync. This requires compatibility test on a variety of large-screen devices and products from mainstream splicer manufacturers in the market. The whole team of more than 40 people repeatedly tested, and the stress test alone was carried out tens of thousands of times.

5.1.2 Main black-box test methods

主要的黑盒测试方法

The methods for black-box test case design mainly include equivalence class division, boundary value analysis, error guess, cause-effect diagram, decision table driven test, orthogonal experiment design, scenario method, and so on. When facing actual software test tasks, using only one black-box test case design method is insufficient to obtain comprehensive test cases. A practical approach is to utilize a combination of test case design techniques in order to improve both the efficiency and coverage of test. This necessitates a thorough understanding of the principles behind these methods and the accumulation of substantial software test experience in order to effectively elevate the level of software test.

5.1.3 Software defects targeted by black-box test

黑盒测试针对的软件缺陷

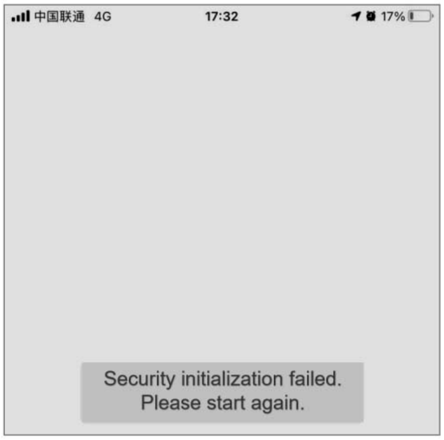
Black-box test can mainly find the following types of errors.

1. Input and output errors

For example, in the user registration interface of an application, there is a text box for entering the user's cell phone number. However, during test it was discovered that the application does not validate the input for a valid cell phone number as required by program specifications. This means that letter input is accepted, which deviates from the intended functionality of receiving a valid cell phone number to send an authentication code later. This failure to perform necessary validation on input data does not align with program specifications and will impact subsequent functions.

2. Initialization, termination error

Initialization error refers to the inability to open the application software normally, as shown in Figure 5-2. A compatibility test of an APP program found that in a particular environment, the APP program will prompt "Security initialization failed. Please start again.", when it is opened after installation.



Security initialization failed.
Please start again.

Figure 5-2 APP initialization failure

Termination errors, such as the APP program always being in a running state after test and execution, but no longer responding to user operations without prompts and unable to exit normally.

3. Incorrect or missing functions

The specification of a student schedule query APP states that it can query the schedule of the current teaching week for students based on their student number. However, black-box test reveals that it can only query the weekly schedule for administrative classes and some elective classes cannot be found, indicating a function omission.

4. Interface Error

During black-box test of a grade management APP, the main interface displays “Welcome to the online bookstore,” which is an information error in the program’s interface.

5. Performance does not meet the requirements

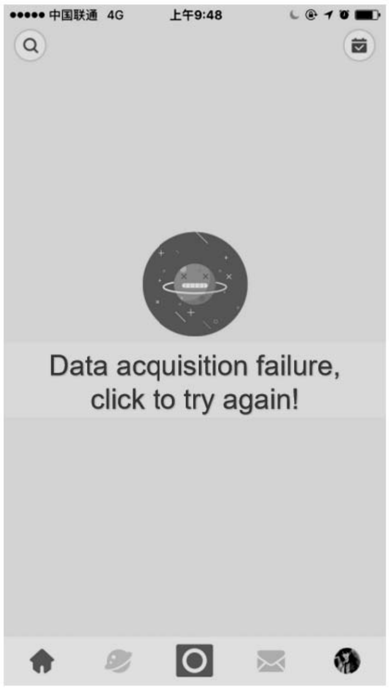
For example, a ticketing APP specifies that it should be able to handle 100,000 mobile customers buying tickets simultaneously. However, black-box test revealed that when simulating 50,000 mobile customers buying tickets at once, the system became paralyzed. This indicates that the system’s performance does not meet requirements.

6. Database or other external data access errors

For example, an application requires access to the underlying database during execution, and during black-box test, it encounters a failure in data retrieval. This indicates a database or other external data access error, as illustrated in Figure 5-3. The cause of this error could be attributed to a weak network connection, system congestion, or programming errors.

7. User privacy, security issues, etc.

As an example, black-box test of a student management application revealed that upon logging into the system with a specific student account, it was possible to view and modify the information of other students. Such a system presents user privacy and security issues that may result in unauthorized information disclosure and tampering.



Data acquisition failure, click to try again!

Figure 5-3 Database or other external data access errors

5.2 Equivalence class division test

等价类划分测试

Theoretically, black-box test can theoretically identify all errors in a program by exhaustively test all possible inputs. This includes not only test all legitimate inputs, but also examining those inputs that are not legitimate but likely to occur. However, exhaustive test is impractical, so it is essential to enhance the focus of the test. This involves not only test a variety of potential scenarios to improve test completeness, but also avoiding duplication and reducing redundancy in order to save on test costs. The equivalence class division test represents one such method for black-box test.

5.2.1 Equivalence class division

等价类划分

What is equivalence class division? Let's consider an example. A school requires uniforms for all students. The school uniform factory takes a sample of the uniforms and asks students to try them on. If there are many students in the school, having everyone try on the uniforms can be very time-consuming and laborious. An efficient approach would be to divide the students into different groups based on their body types, as illustrated in Figure 5-4. In this way, only one student from each group needs to try on the uniform. If it fits one student in a group, then it will also fit the other students in that group because they share the same size. This concept is known as equivalence class division.

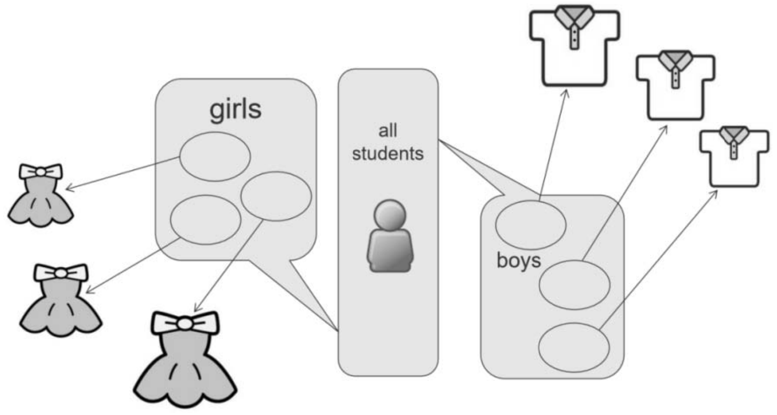


Figure 5-4 Example of equivalence class division

The equivalence class of an element refers to the set of all elements that are equivalent to each other based on a specific equivalence relation. It is a subset of the complete data set, with elements sharing similar characteristics. For instance, integers can be divided into two equivalence classes, odd and even, based on parity.

$$\text{odd-even property}\begin{cases}\text{0, 2, 4, ... even number, are all equivalent.} \\ \text{They form an even equivalence class} \\ \\ \text{1, 3, 5, ... odd number, are all equivalent.} \\ \text{They form an odd equivalence class}\end{cases}$$

When dividing into equivalence classes, it is important to ensure that there are no duplicate elements between two classes and that when combined, all equivalence classes form the entire data set being divided, as depicted in Figure 5-5.

From the perspective of software test, the equivalence class division method assumes that elements within the same equivalence class share similar characteristics and play an equivalent role in discovering or exposing defects in

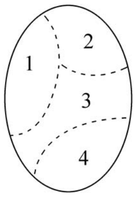


Figure 5-5 Equivalence class division

the program. Therefore, it is reasonable to assume that test one representative data from a certain equivalence class is equivalent to test all data within that class.

In software test, the equivalence class division involves categorizing all possible input data into several equivalence classes and then selecting one or a few representative data from each class to test the program, as illustrated in Figure 5-6.

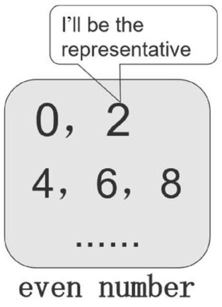


Figure 5-6 For an equivalence class, only a representative of it need be selected for test

By utilizing equivalence class division, we are able to transform potentially infinite inputs into a finite number of equivalence classes. From there, we can select a representative as a test case in order to achieve comprehensive test while minimizing redundancy, reducing costs, and enhancing the effectiveness of the test. Equivalence class division is considered to be the most fundamental and widely used black-box test method.

Equivalence class division test is typically applied to input data based on software specifications. The input data is categorized into different equivalence classes according to various processing methods, and representatives are then chosen from these classes for use as test cases. However, there are instances where equivalence class division test may also be applicable to output data or intermediate process data.

Equivalence classes can be categorized into valid equivalence classes and invalid equivalence classes. A valid equivalence class refers to a collection of input data that is reasonable and meaningful for the program specification. It enables the verification of whether the program meets the predefined features, such as functionality and performance, outlined in the specification. On the other hand, an invalid equivalence class consists of input data that is unreasonable and irrelevant to the program specification. It allows for test whether the program can appropriately handle unusual inputs without undesirable consequences.When designing test cases, it is crucial to consider both types of equivalence classes. This is because software should not only be capable of receiving and processing reasonable data but also resilient enough to withstand unexpected inputs. When faced with unreasonable or irrelevant data inputs, it is essential for the software to handle them properly.

Let's look at an example of the simplest equivalence class division. The symbolic function input x, output y, if x>0, then y =1; if x =0, then y =0; if x<0, then y = −1.

$$\begin{cases} x > 0 \rightarrow y = 1 \\ x = 0 \rightarrow y = 0 \\ x < 0 \rightarrow y = -1 \end{cases}$$

It is not difficult to classify equivalence classes for x. There are three valid equivalence classes for x, x>0, x =0 and x<0.

And the invalid equivalence classes for x can be categorized as all data that cannot be compared with 0.

In the example of the symbolic function, the valid equivalence class of x is divided according to the range, and for different data types and processing rules, the division of equivalence class is not in the same way. The common ways of division are as follows.

① By range.

② By value.

③ By set.

④ By restriction or restriction rule.

⑤ By processing mode, etc.

For example, to test the individual income tax calculation software, in accordance with the individual income tax classification calculation standards, the input data “annual taxable income” is divided into equivalence class by the range, shown as Table 5-1.

Table 5-1 Equivalence class division of “annual taxable income” according to ranges

Equivalence class number	Annual taxable income	Tax rate /%	Quick deduction
1	Not exceeding 36,000 yuan	3	0
2	Exceeding 36,000 to 144,000 yuan	10	2,520
3	Exceeding 144,000 to 300,000 yuan	20	16,920
4	Part exceeding 300,000 to 420,000 Yuan	25	31,920
5	The portion exceeding 420,000 to 660,000 yuan	30	52,920
6	Over 660,000 to 960,000 yuan	35	85,920
7	Exceeding 960,000 yuan	45	181,920

There is a grade processing program for converting five-level points to a 100-point scale. When testing it, the input data can be divided into equivalence classes according to the processing method, shown as Table 5-2.

Table 5-2 Equivalence classes can be divided according to the processing method

Equivalence class number	Five-level points	Processing
1	Excellent	convert to 90
2	Good	convert to 80
3	Moderate	convert to 70
4	Pass	convert to 60
5	Fail	convert to 40

Up to this point, there is no standardized method for classifying equivalence classes with high quality. Different specifications of the software may require different equivalence classes, and the quality of test cases obtained from these classes can vary. When classifying equivalence classes, the following suggestions can be considered.

(1) If the input condition specifies a range of values, then one valid equivalence class and two invalid equivalence classes can be identified.

For example, if the program input condition is an integer x less than or equal to 100, greater than or equal to 0, the valid equivalence class would be 0

$\leqslant x \leqslant 100$ and the two invalid equivalence classes would be $x < 0$ and $x > 100$.

(2) If the input condition specifies a set of input values, then one valid equivalence class and one invalid equivalence class can be determined.

For example, if a program specifies that valid values for the input data job title come from the set R = { Teaching Assistant, Lecturer, Associate Professor, Professor, Other, None }, then the valid equivalence class is that the job title belongs to R and the invalid equivalence class is that the job title does not belong to R.

(3) A valid equivalence class and an invalid equivalence class can be determined if the input condition specifies that the input values must satisfy some requirement.

For example, if a program specifies that the input data x must take a numeric symbol as a condition, then the valid equivalence class is x is a numeric symbol and the invalid equivalence class is x contains a non-numeric symbol.

(4) In the case where the input condition is a boolean quantity, a valid equivalence class and an invalid equivalence class can be determined.

For example, if a program specifies that its valid input is a boolean truth value, the valid equivalence class is the Boolean true value, and the invalid equivalence class is the boolean false value.

(5) If the input data is specified as a set of values (assume n) and the program is to process each input values separately, then n valid equivalence classes and one invalid equivalence class can be determined.

For example, the input of a program comes from the set {excellent, good, medium, pass, fail}, and the program will process these 5 values separately, then there are 5 effective equivalence classes, respectively, x= “excellent”, x= “good”, x= “medium”, x= “pass”, x= “fail”, invalid equivalence class of x is which does not belong to the set {excellent, good, medium, pass, fail}.

(6) If it is stipulated that the input data must conform to certain rules, then it is possible to determine a valid equivalence class (conforming to the rules) and a number of invalid equivalence classes that violate the rules from different perspectives respectively.

For example, a certain message encryption code consists of three parts, the names and contents of which are shown as follows.

Encryption type code : blank or three digits.