Translated English of Chinese Standard: GB/T43499-2023

www.ChineseStandard.net → Buy True-PDF → Auto-delivery.

Sales@ChineseStandard.net

GB

NATIONAL STANDARD OF THE

PEOPLE'S REPUBLIC OF CHINA

ICS 43.180

CCS R 86

GB/T 43499-2023

Testing methods for the software of vehicle inspection system

机动车检测系统软件测试方法

Issued on: December 28, 2023 **Implemented on: July 01, 2024**

Issued by: State Administration for Market Regulation;

National Standardization Administration.

Table of Contents

Testing methods for the software of vehicle inspection system

1 Scope

This document specifies the test content, test methods, and test document set of the motor vehicle inspection agency's inspection system software.

This document is applicable to the testing of inspection system software of motor vehicle inspection agencies.

2 Normative references

The contents of the following documents constitute essential provisions of this document through normative references in the text. Among them, for dated reference documents, only the version corresponding to the date applies to this document; for undated reference documents, the latest version (including all amendments) applies to this document.

GB/T 25000.51-2016 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Part 51: Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing

GB/T 26765 Specifications for motor vehicle safety inspection business information system and networking

GB/T 38634.4-2020 Systems and software engineering - Software testing - Part 4: Test techniques

GB/T 42685 Terminology of power-driven vehicle inspection

HJ 1238 Technical specification for data collection and transmission of vehicles periodic emissions inspection

3 Terms and definitions

The terms and definitions defined in GB/T 42685 and the following apply to this document.

3.1

Vehicle inspection institution

Test the extent to which an activity or event can be proven and undeniable after it has occurred. Data non-repudiation testing includes the following:

a) Function to provide evidence to the data originator: Test whether the software has the function to provide evidence of the origin of the data to the data originator upon request;

b) Function to provide evidence to the data recipients: Test whether the software has the function of providing data recipients with evidence of data receipt upon request.

4.2.2.3 Data verifiability

Test the extent to which an entity's activities can be uniquely traced to that entity. Data verifiability testing includes the following:

a) User process association and traceability: Test whether the software can associate the user process with the owner user, so that the behavior of the user process can be traced back to the owner user of the process;

b) Process dynamic association and traceability: Test whether the software can associate the system process dynamics with the current service requester user, so that the behavior of the system process can be traced back to the current service requester user.

4.2.2.4 Data authenticity

Test the extent to which the identity of a test object or resource can be verified to be consistent with its claims. Data authenticity testing includes the following:

a) User list and configuration table: Test whether the software has the user list and configuration table of the current system;

b) Access login record: Test whether the access login record of the software is complete in the system's access history database;

c) Historical logs and log management: Test whether the software has historical logs and log management functions for users to use the system;

d) Simulated intrusion log records: Test whether the software log content has relevant records when the software is invaded by a simulated attack event;

e) Virus detection records: Test whether the records of software users' access to systems and data include "virus detection records" to prevent viruses.

4.2.2.5 Data transmission security

Test how well data is protected during transmission and processing. Data transmission

security testing includes the following:

a) Verification: Test whether the data verification code algorithm is used, generate the verification code of the source data, verify the integrity of important data during transmission and processing to prevent key data from being illegally tampered with;

b) Data encryption: Test whether encryption technology is used to encrypt important data and private information to protect data confidentiality and prevent information leakage;

c) Network transmission security: Test whether measures are taken to ensure the security of data transmission between different networks.

4.2.3 Source code standardization

Check the program and discover possible exceptions in the program. Source code testing includes but is not limited to the following:

a) Statement label does not exist: Test whether there is a statement label that redirects to a statement label that does not exist;

b) Unused statement labels: Test whether there are unused statement labels;

c) Unused subroutine definitions: Test whether there are unused subroutine definitions;

d) Subroutine does not exist: Test whether to call a subroutine that does not exist;

e) Unreachable statements: Test whether there are statements that cannot be reached after entering from the program entrance;

f) Statements that cannot reach the stop statement: Test whether there are statements that cannot reach the stop statement;

g) Special trigger pop-up window: Test whether there are special trigger conditions and a pop-up window will appear.

4.2.4 Others

Check software version registration, upgrades, changes and other records. Other tests include but are not limited to the following:

a) Registration form: Check whether the software has a version registration form, change and upgrade record form (if it has been changed or upgraded), etc.;

b) Registration form and software consistency: Check the version registration form, upgrade (change) internal approval form, upgrade (change) record form, etc., to

Equivalence class division uses the test item model to divide test item inputs and outputs into equivalence classes (also called "partitions"), where each equivalence class shall be used as a test condition. These equivalence classes shall be derived from the test basis such that all values in each partition can be treated similarly by the test item (i.e., the values in the equivalence class are "equivalent"). Valid inputs and output as well as invalid input and output can derive equivalence class division.

Each equivalence class shall be a test coverage item (that is, in the equivalence class division, the test conditions and test coverage items are the same equivalence class). Exported test cases shall implement each test coverage item (i.e., equivalence class). The steps to export test cases are as follows.

a) Determine the combination method of test coverage achieved by selecting test cases. The following are two common methods:

   1) One-to-one, each exported test case is used to cover a specific equivalence class;

   2) Minimization, where equivalence classes are covered by test cases such that the minimum number of test cases derived covers all equivalence classes at least once.

b) Use the method in step a) to select the test coverage items included in the current test case.

c) Determine the input values for the test coverage items covered by the test case for execution, as well as any valid values for any other input variables required by the test case.

d) Apply the inputs to the test basis, to determine the expected results of the test case.

e) Repeat steps b) ~ d) until the required test coverage is achieved.

5.2.2 Classification tree

The classification tree method uses the test item model to divide the input of the test item and graphically represents it in the form of a classification tree. The input of the test item is divided into several "categories"; each division consists of several independent (non-overlapping) "categories" and subcategories; meanwhile the category set is complete (all input domains of the modeled test item are recognized and included in all categories). Each category shall be a test condition. Depending on the rigor of the test, the "category" obtained by decomposing the classification may be further divided into "subcategories". Depending on the required test coverage, the exported partitions and categories may include both valid and invalid input data. Shape the hierarchical relationship between classification, category, subcategories into a tree, using the input domain of the test item as the root node of the tree, the classification as the branch node, the category, subcategories as the leaf node.

Test coverage items shall be derived from the combination classification using the selected combination method; the exported test cases shall implement each test coverage item. The steps to export test cases are as follows:

a) Export test coverage items; select a combination for the current test case, requiring that the combination is not covered by test cases;

b) Determine the input values in each category that have not yet been assigned a value;

c) Determine the expected results of the test case by applying the inputs to the test basis;

d) Repeat steps a) ~ c), until the required test coverage level is reached.

5.2.3 Boundary value analysis

Boundary value analysis divides the input and output of the test item into multiple ordered sets and subsets (partitions and sub-partitions) with identifiable boundaries, through the analysis of the boundary value of the test item model, where each boundary is a test condition. Boundaries shall be derived from the test basis.

The exported test cases shall implement each test coverage item. Below are the steps to export test cases.

a) To determine the combination of test coverage items achieved by selecting test cases, there are two common methods:

   1) One-to-one, each test case implements a specified boundary value;

   2) Minimization, which derives a minimum number of test cases to cover all boundary values at least once.

b) Use the method in step a) to select the test coverage items included in the current test case.

c) Other input variables not selected by the test case in step b) take any valid value.

d) Determine the expected results of the test case by applying the inputs to the test evidence.

e) Repeat steps b) ~ d), until the required test coverage level is reached.

5.2.4 Cause-and-effect diagram

The cause-and-effect diagram method uses a cause-and-effect diagram to represent the logical relationship model between the cause (such as input) and the result (such as output) of the test item, including:

b) Determine the test case input values that cover the test coverage items;

c) Determine the expected results of the test case by applying the inputs to the test basis (the expected results can be defined using the access states described in the output and state models);

d) Repeat steps a) ~ c), until the required test coverage level is reached.

5.2.6 Scenario testing

Scenario testing uses a sequence model of interactions between a test item and other systems (in this context, users are often considered to be other systems), to test the involved test item usage processes. The test condition shall be one interaction sequence (i.e., one scenario) or all interaction sequences (i.e., all scenarios).

Scenario testing shall include the following scenarios:

The "main" scenario is the expected typical action sequence of the test item, or an arbitrary choice taken where there is no typical action sequence; the "alternative" scenario represents the optional (non-main) scenario of the test item.

Test coverage items shall be the main scenario and alternative scenarios (that is, the test coverage items are the same as the test conditions). Among the test cases exported by scenario testing, one test case covers at least one scenario (test coverage item). The steps to export test cases are as follows:

a) Select the test coverage items implemented by the current test case;

b) Determine the input values of test coverage items covered by test cases;

c) Determine the expected results of the test case by applying the input to the test basis;

d) Repeat steps a) ~ c), until the required test coverage level is reached.

5.2.7 Random testing

Random testing uses an input domain model of the test item to define the set of all possible input values. The input distribution shall be chosen to generate random input values. The entire input field shall be randomly tested against test conditions.

Random testing has no known test coverage items; test cases for random testing shall randomly select input values from the input field of the test item (or pseudo-randomly if a tool is used), according to the selected input distribution. The steps to export test cases are as follows:

a) Select an input distribution for the test input;

b) Generate random values of the test input according to the input distribution in step a);

c) Determine the expected results of the test case by applying the input to the test basis;

d) Repeat steps b) ~ c), until the required tests are completed.

5.3 Structure-based testing methods

5.3.1 Statement test

Statement testing shall export the source code model of the test item and identify the statement as executable or non-executable. Each execution statement shall be a test condition.

The steps to export test cases are as follows:

a) Identify control flow subpaths that reach one or more test coverage items that have not yet been executed to test coverage;

b) Determine test inputs that implement the identified control flow subpaths;

c) Determine the expected results of executing the control flow subpath by applying the corresponding test inputs to the test basis;

d) Repeat steps a) ~ c), until the required test coverage level is reached.

5.3.2 Branch testing

Branch testing shall derive a test item control flow model that identifies control flow branches. Each branch of the control flow model is a test condition. The branches include:

- Conditional transition from any node in the control flow to another node;

- An explicit and unconditional transition of control from any node in the flow of control to another node;

- When a test item has more than one entry point, control transition to one entry point of the test item.

Each branch in the control flow model is a test coverage item (that is, the test coverage item is the same as the test condition). The steps to export test cases are as follows:

a) Identify control flow subpaths that reach one or more test coverage items that have not yet been executed to test coverage;

d) Determine the expected results of executing the control flow subpath by applying the corresponding test inputs to the test basis;

e) Repeat steps a) ~ d), until the required test coverage level is reached.

5.3.5 Branch condition combination test

Branch condition combination testing shall derive a test item control flow model that identifies decisions and conditions. In branch condition combination testing, each decision shall be a test condition.

Each unique possible combination of Boolean values for the conditions in each decision shall be identified as a test coverage item. It includes simple judgments, that is, a judgment result as formed by the combination of two single Boolean quantities. The steps to export test cases are as follows:

a) Identify control flow subpaths that reach one or more test coverage items that have not yet been executed to test coverage;

b) Determine test inputs that implement the identified control flow subpaths;

c) Identify a subset of the test inputs in step b), to cover selected combinations of Boolean values for the conditions included in the decision;

d) Determine the expected results of executing the control flow subpath, by applying the corresponding test inputs to the test basis;

e) Repeat steps a) ~ d), until the required test coverage level is reached.

5.3.6 Modified condition decision coverage test

Modified condition decision coverage tests shall derive a test item control flow model that identifies decisions and conditions. In modified condition decision coverage (MCDC) testing, each decision shall be a test condition.

A decision condition that can independently affect the decision result by a single Boolean condition; each unique feasible combination of its Boolean values serves as a test coverage item. Indicate whether a condition independently affects a decision result by changing only one condition while keeping other possible conditions unchanged. The following steps shall be followed while exporting test cases:

a) Identify control flow subpaths that reach one or more test coverage items that have not yet been executed to test coverage;

b) Determine the test inputs that enable the identified control flow subpaths to be executed;

c) Determine the test input subset in step b) to cover the selected combinations of

each Boolean value in the determination. Each combination condition can independently affect the determination result, allowing a single Boolean condition to independently affect the determination result;

d) Determine expected results by applying corresponding test inputs to the test basis;

e) Repeat steps a) ~ d), until the required test coverage level is reached.

5.3.7 Data flow testing

5.3.7.1 Full definition testing

Test coverage is the control flow subpath, from each variable definition to some use of that definition (either a predicate use or a computation use). Each subpath is called a define-use path. The "full definition" test requires that all variable definitions cover a subpath from definition to its predicate use or computed use from at least one definition to any type of use (relevant to the specific variable). The following steps shall be followed while exporting test cases:

a) Identify definitions not yet covered by tests;

b) Determine the test input of the control flow subpath obtained from the definition to be executed;

c) Determine the expected results of executing the control flow subpath by applying the corresponding test inputs to the test basis;

d) Repeat steps a) ~ c) until the required test coverage level is reached.

5.3.7.2 Full computing usage test

Test coverage is the control flow subpath, from each variable definition to each calculation used by that definition. The "full computing usage" test requires that all relevant variable definitions cover at least one freely defined subpath (relevant to a specific variable), from the definition to each of its computing uses. The following steps shall be followed while exporting test cases:

a) Determine the control flow subpath (excluding intermediate definitions) used from the variable definition to the definition calculation, which has not been covered by the test;

b) Determine the test input of the control flow subpath to be executed;

c) Determine the expected results of executing the control flow subpath, by applying the corresponding inputs to the test basis;

d) Repeat steps a) ~ c), until the required test coverage level is reached.

b) Determine the test input of the control flow sub-path to be executed;

c) Determine the expected results of executing the control flow subpath, by applying the test inputs to the test basis;

d) Repeat steps a) ~ c), until the required test coverage level is reached.

5.4 Experience-based testing methods

Experience-based testing method, also known as error guessing method. The error-guessing method involves designing a checklist of the types of defects that may exist in a test item, allowing the tester to identify test item inputs that could cause a failure if those defects were present in the test item. Each defect type shall be tested as a condition.

There is no generally accepted test coverage for the error guessing method. Test cases for the error guessing method are usually created by selecting the defect type from a checklist of defect types to be covered and exporting test cases that detect the defect type (if present) in the test item. When exporting test cases, it shall use the following step:

a) Select the defect type covered by the current test case;

b) Determine the input values that may cause a failure corresponding to the selected defect type;

c) Determine the expected results of the test case, by applying the inputs to the test basis;

d) Repeat steps a) ~ c), until the required tests are completed.

6 Test document set

6.1 General

The requirements for the test document set are in accordance with the provisions of GB/T 25000.51-2016. The test document set mainly includes test plans, test cases, test records, test reports.

6.2 Test plan

The test plan describes the tests to be completed, including test background, test purpose, test content, required resources, task arrangement and progress, etc. The test plan shall at least include the following contents:

a) Develop testing content, strategies, methods for each scope;

b) Standards used for testing;

This is an excerpt of the PDF (Some pages are marked off intentionally)

Full-copy PDF can be purchased from 1 of 2 websites:

1. https://www.ChineseStandard.us

- SEARCH the standard ID, such as GB 4943.1-2022.
- Select your country (currency), for example: USA (USD); Germany (Euro).
- Full-copy of PDF (text-editable, true-PDF) can be downloaded in 9 seconds.
- Tax invoice can be downloaded in 9 seconds.
- Receiving emails in 9 seconds (with download links).

2. https://www.ChineseStandard.net

- SEARCH the standard ID, such as GB 4943.1-2022.
- Add to cart. Only accept USD (other currencies - https://www.ChineseStandard.us).
- Full-copy of PDF (text-editable, true-PDF) can be downloaded in 9 seconds.
- Receiving emails in 9 seconds (with PDFs attached, invoice and download links).

Translated by: Field Test Asia Pte. Ltd. (Incorporated & taxed in Singapore. Tax ID: 201302277C)

About Us (Goodwill, Policies, Fair Trading…): https://www.chinesestandard.net/AboutUs.aspx

Contact: Wayne Zheng, Sales@ChineseStandard.net

Linkin: https://www.linkedin.com/in/waynezhengwenrui/

------ The End ------